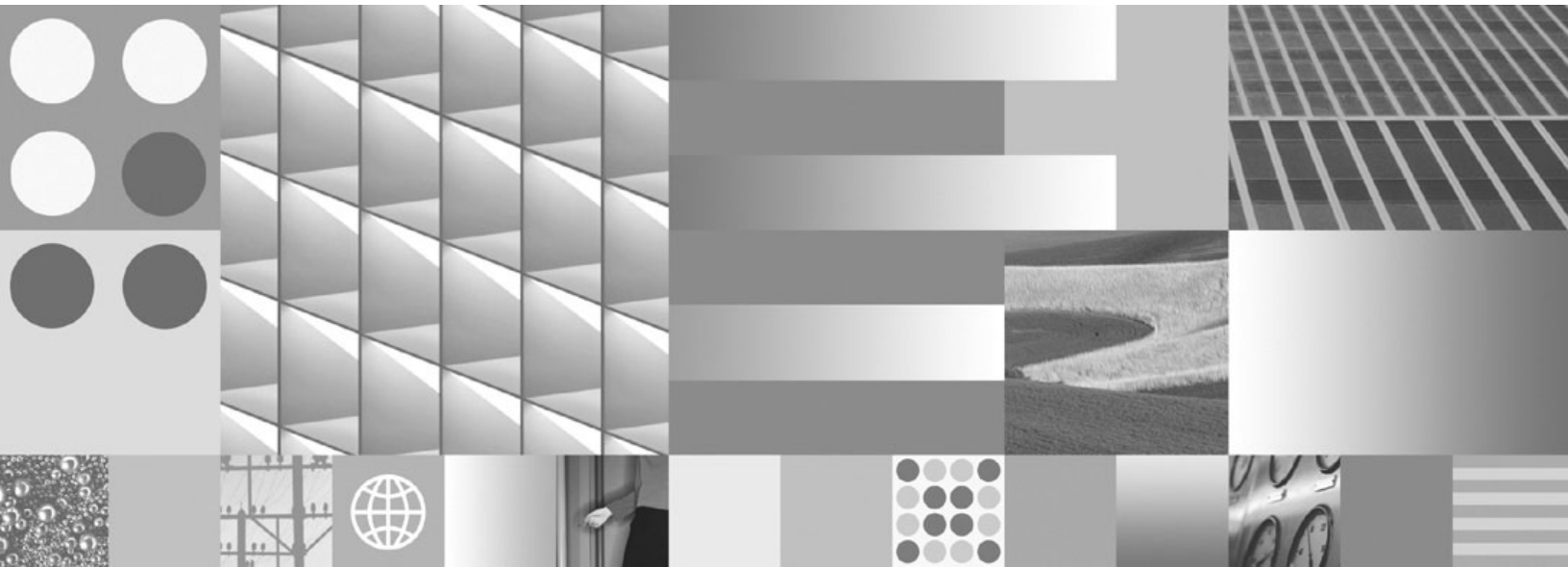




**Spatial Extender and Geodetic Data Management Feature Guía del usuario y manual de consulta**





**Spatial Extender and Geodetic Data Management Feature Guía del usuario y manual de consulta**

**Nota**

Antes de utilizar esta información y el producto al que da soporte, lea la información general contenida en el apartado Apéndice B, "Avisos", en la página 517.

**Nota de edición**

Este manual es la traducción del original en inglés *IBM DB2 9.7 for Linux, UNIX, and Windows Spatial Extender and Geodetic Data Management Feature User's Guide and Reference* (SC27-2468-00).

Este documento contiene información propiedad de IBM. Se proporciona según un acuerdo de licencia y está protegido por la ley de la propiedad intelectual. La información contenida en esta publicación no incluye ninguna garantía de producto, por lo que ninguna declaración proporcionada en este manual deberá interpretarse como tal.

Puede realizar pedidos de publicaciones de IBM en línea o a través del representante de IBM de su localidad.

- Para realizar pedidos en línea, vaya a IBM Publications Center ubicado en el sitio web [www.ibm.com/shop/publications/order](http://www.ibm.com/shop/publications/order)
- Para encontrar al representante de IBM de su localidad, vaya al IBM Directory of Worldwide Contacts en el sitio web [www.ibm.com/planetwide](http://www.ibm.com/planetwide)

Para realizar pedidos de publicaciones de DB2 desde DB2 Marketing and Sales, en los EE.UU. o en Canadá, llame al 1-800-IBM-4YOU (426-4968).

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo a utilizar o distribuir dicha información en la forma en que IBM considere adecuada, sin contraer por ello ninguna obligación con el remitente.

© Copyright International Business Machines Corporation 1998, 2009.

# Contenido

## Capítulo 1. Acerca de DB2 Spatial

<b>Extender</b> . . . . .	<b>1</b>
La finalidad de DB2 Spatial Extender . . . . .	1
Cómo los datos representan elementos geográficos . . . . .	2
Naturaleza de los datos espaciales . . . . .	3
Naturaleza de los datos geodésicos . . . . .	4
De dónde proceden los datos espaciales . . . . .	4
Cómo están interrelacionados los elementos geográficos, la información espacial, los datos espaciales y las geometrías. . . . .	6

## Capítulo 2. Acerca de las geometrías . . . 9

Geometrías . . . . .	9
Propiedades de las geometrías . . . . .	11
Tipo . . . . .	11
Las coordenadas de la geometría . . . . .	11
Coordenadas X e Y. . . . .	12
Coordenadas Z . . . . .	12
Coordenadas M . . . . .	12
Interior, perímetro y exterior. . . . .	12
Simple o no simple. . . . .	12
Cerrada . . . . .	12
Vacía o no vacía . . . . .	12
Rectángulo delimitador mínimo (MBR) . . . . .	12
Dimensión. . . . .	13
Identificador de sistemas de referencia espacial . . . . .	13

## Capítulo 3. Cómo utilizar DB2 Spatial

<b>Extender</b> . . . . .	<b>15</b>
Cómo utilizar DB2 Spatial Extender . . . . .	15
Interfaces para DB2 Spatial Extender y funciones asociadas . . . . .	15
Tareas que el usuario realiza para configurar DB2 Spatial Extender y crear proyectos. . . . .	15

## Capítulo 4. Iniciación a DB2 Spatial

<b>Extender</b> . . . . .	<b>21</b>
Configuración e instalación de Spatial Extender . . . . .	21
Requisitos del sistema para instalar Spatial Extender . . . . .	22
Instalación de DB2 Spatial Extender (Windows) . . . . .	22
Instalación de Spatial Extender mediante el asistente de instalación . . . . .	23
Instalación de Spatial Extender mediante un archivo de respuestas . . . . .	23
Instalación de DB2 Spatial Extender (Linux, UNIX) . . . . .	23
Instalación de DB2 Spatial Extender utilizando el asistente de instalación de DB2 (Linux, UNIX). . . . .	24
Instalación de Spatial Extender mediante el mandato db2_install (Linux, UNIX) . . . . .	24
Instalación de Spatial Extender mediante un archivo de respuestas . . . . .	24
Verificación de la instalación de Spatial Extender . . . . .	24
Consideraciones posteriores a la instalación. . . . .	26
Descarga de ArcExplorer para DB2 . . . . .	26

## Capítulo 5. Actualización a DB2 Spatial

<b>Extender Versión 9.7.</b> . . . . .	<b>27</b>
Actualización de DB2 Spatial Extender . . . . .	27
Actualización de DB2 Spatial Extender . . . . .	28

## Capítulo 6. Configuración de una base de datos . . . . . 29

Configuración de una base de datos para alojar datos espaciales . . . . .	29
Ajuste de las características de anotaciones cronológicas de transacción . . . . .	29
Ajuste del tamaño de la pila de aplicaciones . . . . .	31

## Capítulo 7. Configuración de recursos espaciales para una base de datos . . . 33

Cómo configurar recursos en la base de datos . . . . .	33
Inventario de recursos suministrados para la base de datos . . . . .	33
Habilitación de una base de datos para operaciones espaciales . . . . .	34
Cómo trabajar con datos de referencia . . . . .	34
Datos de referencia . . . . .	34
Configuración del acceso a los datos de referencia para DB2SE_USA_GEOCODER . . . . .	35
Registro de un geocodificador . . . . .	35

## Capítulo 8. Configuración de recursos espaciales para un proyecto . . . . . 37

Cómo utilizar sistemas de coordenadas . . . . .	37
Sistemas de coordenadas . . . . .	37
Sistema de coordenadas geográficas . . . . .	37
Sistemas de coordenadas proyectadas . . . . .	43
Selección o creación de sistemas de coordenadas . . . . .	44
Cómo configurar sistemas de referencia espacial . . . . .	45
Sistemas de referencia espacial . . . . .	45
Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo . . . . .	47
Sistemas de referencia espacial suministrados con DB2 Spatial Extender . . . . .	48
Factores de conversión que transforman datos de coordenadas en enteros . . . . .	50
Creación de un sistema de referencia espacial . . . . .	52
Cálculo de los factores de escala . . . . .	53
Factores de conversión que transforman datos de coordenadas en enteros . . . . .	54
Determinación de las coordenadas y medidas mínimas y máximas . . . . .	54
Cálculo de los valores de desplazamiento . . . . .	55
Creación de un sistema de referencia espacial . . . . .	56

## Capítulo 9. Configuración de columnas espaciales . . . . . 59

Columnas espaciales . . . . .	59
-------------------------------	----

Columnas espaciales con contenido visualizable	59
Tipos de datos espaciales	59
Creación de columnas espaciales	61
Registro de columnas espaciales	62

## Capítulo 10. Cómo llenar columnas espaciales . . . . . 65

Acerca de la importación y de la exportación de datos espaciales	65
Importación de datos espaciales	66
Importación de datos de formas a una tabla nueva o existente	66
Exportación de datos espaciales	67
Exportación de datos a un archivo de formas	67
Cómo utilizar un geocodificador	68
Geocodificadores y geocodificación	68
Definición de las operaciones de geocodificación	70
Configuración de un geocodificación para que se ejecute automáticamente	72
Ejecución de un geocodificador en modalidad de proceso por lotes	73

## Capítulo 11. Utilización de índices y vistas para acceder a datos espaciales. 75

Tipos de índices espaciales	75
Índices reticulares espaciales.	75
Generación de índices reticulares espaciales	76
Uso de funciones espaciales en una consulta	76
Cómo una consulta utiliza un índice reticular espacial.	77
Consideraciones sobre el número de niveles de índice y tamaños de retícula	78
Número de niveles de índice	78
Tamaños de las celdas reticulares	78
Creación de índices reticulares espaciales	82
Creación de un índice espacial utilizando SQL CREATE INDEX.	83
Sentencia CREATE INDEX para un índice reticular espacial.	84
Ajuste de índices reticulares espaciales con Index Advisor	85
Ajuste de índices reticulares espaciales con Index Advisor—Visión general	85
Determinación de los tamaños de retícula para un índice reticular espacial	85
Análisis de estadísticas de índices reticulares espaciales	87
El mandato gseidx	91
Utilización de vistas para acceder a columnas espaciales	94

## Capítulo 12. Análisis y generación de información espacial. . . . . 95

Entornos para realizar análisis espaciales	95
Ejemplos del funcionamiento de las funciones espaciales	95
Funciones que utilizan índices para optimizar consultas	96

## Capítulo 13. Mandatos de DB2 Spatial Extender . . . . . 99

Invocación de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos	99
Mandato db2se upgrade.	104
Mandato db2se migrate	106
Mandato db2se restore_indexes	107
Mandato db2se save_indexes	108

## Capítulo 14. Escritura de aplicaciones y utilización del programa de ejemplo . 111

Cómo incluir el archivo de cabecera de DB2 Spatial Extender en aplicaciones espaciales	111
Cómo llamar a los procedimientos almacenados de DB2 Spatial Extender desde una aplicación	111
Programa de ejemplo de DB2 Spatial Extender	113

## Capítulo 15. Identificación de problemas de DB2 Spatial Extender. . 121

Cómo interpretar mensajes de DB2 Spatial Extender	121
Parámetros de salida de procedimientos almacenados de DB2 Spatial Extender	123
Mensajes de las funciones de DB2 Spatial Extender	125
Mensajes del CLP de DB2 Spatial Extender	126
Mensajes del Centro de control de DB2.	128
Rastreo de problemas de DB2 Spatial Extender con el mandato db2trc.	129
El archivo de notificaciones de administración	130

## Capítulo 16. DB2 Geodetic Data Management Feature . . . . . 133

DB2 Geodetic Data Management Feature	133
Cuándo se utiliza DB2 Geodetic Data Management Feature y cuándo se utiliza DB2 Spatial Extender	133
Sistemas de referencia geodésicos.	134
Latitud y longitud geodésica	135
Distancias geodésicas.	136
Regiones geodésicas	137

## Capítulo 17. Configuración de DB2 Geodetic Data Management Feature . 139

Configuración y habilitación de DB2 Geodetic Data Management Feature	139
Migración de Informix Geodetic DataBlade a DB2 Geodetic Data Management Feature.	140
Cómo llenar datos espaciales con datos geodésicos	146

## Capítulo 18. Índices geodésicos . . . . 147

Índices Voronoi geodésicos	147
Estructuras de celdas de Voronoi	147
Consideraciones para seleccionar una estructura de celdas de Voronoi alternativa	149
Creación de índices Voronoi geodésicos	149
Sentencia CREATE INDEX para un índice Voronoi geodésico.	150
Estructuras de celdas Voronoi suministradas con DB2 Geodetic Data Management Feature	152

Mundo, basado en la densidad de población (ID de Voronoi: 1) . . . . .	153
Estados Unidos (ID de Voronoi: 2) . . . . .	154
Canadá (ID de Voronoi: 3) . . . . .	156
India (ID de Voronoi: 4) . . . . .	157
Japón (ID de Voronoi: 5) . . . . .	158
África (ID de Voronoi: 6) . . . . .	159
Australia (ID de Voronoi: 7) . . . . .	159
Europa (ID de Voronoi: 8) . . . . .	161
Norteamérica (ID de Voronoi: 9) . . . . .	161
Sudamérica (ID de Voronoi: 10) . . . . .	162
Mediterráneo (ID de Voronoi: 11) . . . . .	163
Mundo, distribución uniforme de datos, resolución media: dodeca04 (ID de Voronoi: 12) .	164
Mundo, países industrializados: países del G7 (ID de Voronoi: 13) . . . . .	165
Mundo, distribución uniforme de datos, resolución baja: isotipo (ID de Voronoi: 14) . . . . .	166

## Capítulo 19. Diferencias en la utilización de datos geodésicos y espaciales. . . . . 169

Atributos x e y mínimos y máximos. . . . .	169
Diferencias entre la utilización de representaciones terrestres y planas . . . . .	169
Segmentos lineales que cruzan el meridiano número 180 . . . . .	170
Polígonos que ocupan el meridiano 180° . . . . .	171
Polígonos que delimitan un polo . . . . .	174
Polígonos que representan hemisferios, cinturones ecuatoriales y toda la tierra . . . . .	175
Funciones espaciales soportadas por DB2 Geodetic Data Management Feature . . . . .	179
Procedimientos almacenados y vistas de catálogo de DB2 Geodetic Data Management Feature . . . . .	183
Sistemas de referencia soportados por DB2 Geodetic Data Management Feature . . . . .	184
Esferoides geodésicos. . . . .	191

## Capítulo 20. Procedimientos almacenados . . . . . 193

ST_alter_coordsys . . . . .	194
ST_alter_srs . . . . .	196
ST_create_coordsys . . . . .	200
ST_create_srs . . . . .	202
ST_disable_autogeocoding . . . . .	208
ST_disable_db . . . . .	210
ST_drop_coordsys . . . . .	212
ST_drop_srs . . . . .	213
ST_enable_autogeocoding . . . . .	214
ST_enable_db . . . . .	216
ST_export_shape . . . . .	218
ST_import_shape . . . . .	221
ST_register_geocoder . . . . .	229
ST_register_spatial_column . . . . .	234
ST_remove_geocoding_setup . . . . .	235
ST_run_geocoding . . . . .	237
ST_setup_geocoding . . . . .	240
ST_unregister_geocoder . . . . .	244
ST_unregister_spatial_column . . . . .	245

## Capítulo 21. Vistas de catálogo. . . . . 249

Vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS . . . . .	249
Vista de catálogo DB2GSE.SPATIAL_REF_SYS . . . . .	250
Vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS . . . . .	251
Vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS . . . . .	252
Vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS . . . . .	253
Vista de catálogo DB2GSE.ST_GEOCODERS . . . . .	254
Vista de catálogo DB2GSE.ST_GEOCODING . . . . .	255
Vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS . . . . .	256
Vista de catálogo DB2GSE.ST_SIZINGS . . . . .	257
Vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS . . . . .	258
Vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE . . . . .	261

## Capítulo 22. Funciones espaciales: categorías y usos . . . . . 263

Funciones espaciales: categorías y usos . . . . .	263
Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos . . . . .	263
Funciones de Constructor . . . . .	263
Funciones que operan sobre formatos de intercambio de datos . . . . .	264
Una función que crea geometrías a partir de coordenadas . . . . .	265
Ejemplos . . . . .	266
Conversión a una representación de texto convencional (WKT) . . . . .	267
Conversión a una representación binaria convencional (WKB) . . . . .	269
Conversión a la representación de forma ESRI . . . . .	270
Conversión a la representación GML (Geography Markup Language) . . . . .	270
Funciones que comparan elementos geográficos . . . . .	271
Funciones de comparación . . . . .	272
Funciones de comparación espaciales . . . . .	273
Funciones que comprueban si una geometría contiene otra . . . . .	274
ST_Contains . . . . .	274
ST_Within . . . . .	275
Funciones que comprueban intersecciones entre geometrías . . . . .	277
ST_Intersects . . . . .	277
ST_Crosses . . . . .	278
ST_Overlaps . . . . .	279
ST_Touches . . . . .	280
Funciones que comparan envolturas de geometrías . . . . .	282
ST_EnvIntersects . . . . .	282
ST_MBRIntersects . . . . .	282
Funciones que comprueban si dos elementos son idénticos . . . . .	282
ST_EqualCoordsys . . . . .	282
ST_Equals . . . . .	282
ST_EqualSRS . . . . .	283

Función que comprueba si no hay intersección entre dos geometrías . . . . .	284	Funciones que identifican el sistema de referencia espacial de una geometría . . . . .	291
Función que compara geometrías con la serie de la matriz patrón DE-9IM . . . . .	285	ST_SrsId (también llamada ST_SRID) . . . . .	291
Funciones que devuelven información sobre propiedades de geometrías . . . . .	285	ST_SrsName. . . . .	291
Función que devuelve información sobre el tipo de datos . . . . .	285	Funciones que generan nuevas geometrías a partir de las existentes . . . . .	291
Funciones que devuelven información sobre coordenadas y medidas . . . . .	285	Funciones que convierten una geometría en otra	292
ST_CoordDim . . . . .	286	ST_Polygon . . . . .	292
ST_IsMeasured . . . . .	286	ST_ToGeomColl . . . . .	292
ST_IsValid . . . . .	286	ST_ToLineString . . . . .	292
ST_Is3D . . . . .	286	ST_ToMultiLine . . . . .	292
ST_M . . . . .	286	ST_ToMultiPoint . . . . .	292
ST_MaxM . . . . .	286	ST_ToMultiPolygon . . . . .	292
ST_MaxX. . . . .	286	ST_ToPoint . . . . .	292
ST_MaxY. . . . .	286	ST_ToPolygon . . . . .	292
ST_MaxZ. . . . .	287	Funciones que crean nuevas geometrías con configuraciones espaciales diferentes . . . . .	293
ST_MinM. . . . .	287	ST_Buffer. . . . .	293
ST_MinX. . . . .	287	ST_ConvexHull. . . . .	294
ST_MinY. . . . .	287	ST_Difference . . . . .	294
ST_MinZ. . . . .	287	ST_Intersection . . . . .	295
ST_X . . . . .	287	ST_SymDifference . . . . .	296
ST_Y . . . . .	287	Funciones que obtienen una geometría a partir de varias . . . . .	296
ST_Z . . . . .	287	MBR Aggregate (Agregado MBR) . . . . .	296
Funciones que devuelven información sobre geometrías contenidas en una geometría . . . . .	287	ST_Union. . . . .	297
ST_Centroid . . . . .	288	Union Aggregate (Agregado de unión) . . . . .	297
ST_EndPoint . . . . .	288	Funciones que trabajan con medidas . . . . .	297
ST_GeometryN . . . . .	288	ST_DistanceToPoint . . . . .	297
ST_LineStringN . . . . .	288	ST_FindMeasure o ST_LocateAlong . . . . .	298
ST_MidPoint . . . . .	288	ST_MeasureBetween o ST_LocateBetween . . . . .	300
ST_NumGeometries . . . . .	288	ST_PointAtDistance . . . . .	301
ST_NumLineStrings . . . . .	288	Funciones que crean formas modificadas de geometrías existentes . . . . .	302
ST_NumPoints . . . . .	288	ST_AppendPoint . . . . .	303
ST_NumPolygons . . . . .	288	ST_ChangePoint . . . . .	303
ST_PointN . . . . .	288	ST_Generalize . . . . .	303
ST_PolygonN . . . . .	289	ST_M . . . . .	303
ST_StartPoint . . . . .	289	ST_PerpPoints . . . . .	303
Funciones que proporcionan información sobre perímetros, envolturas y anillos . . . . .	289	ST_RemovePoint . . . . .	303
ST_Envelope . . . . .	289	ST_X . . . . .	303
ST_EnvIntersects . . . . .	289	ST_Y . . . . .	303
ST_ExteriorRing . . . . .	289	ST_Z . . . . .	304
ST_InteriorRingN . . . . .	289	Función que devuelve información sobre distancias	304
ST_MBR . . . . .	290	Función que proporciona información sobre índices	304
ST_MBRIntersects . . . . .	290	Conversiones entre sistemas de coordenadas . . . . .	304
ST_NumInteriorRing . . . . .	290	<b>Capítulo 23. Funciones espaciales:</b>	
ST_Perimeter . . . . .	290	<b>sintaxis y parámetros . . . . .</b>	<b>305</b>
Funciones que devuelven información sobre las dimensiones de una geometría . . . . .	290	Funciones espaciales: consideraciones y tipos de datos asociados. . . . .	305
ST_Area . . . . .	290	Factores a tener en cuenta . . . . .	306
ST_Dimension . . . . .	290	Tratamiento de valores de ST_Geometry como valores de un subtipo . . . . .	306
ST_Length . . . . .	290	Funciones espaciales de acuerdo con el tipo de datos de entrada . . . . .	307
Funciones que indican si una geometría es cerrada, vacía o simple . . . . .	290	EnvelopesIntersect. . . . .	309
ST_IsClosed . . . . .	291	MBR Aggregate (Agregado MBR). . . . .	311
ST_IsEmpty . . . . .	291	ST_AppendPoint . . . . .	313
ST_IsSimple . . . . .	291	ST_Area . . . . .	314
		ST_AsBinary . . . . .	317



ST_AsGML . . . . .	319
ST_AsShape . . . . .	320
ST_AsText . . . . .	321
ST_Boundary . . . . .	322
ST_Buffer . . . . .	324
ST_Centroid . . . . .	327
ST_ChangePoint . . . . .	328
ST_Contains . . . . .	329
ST_ConvexHull . . . . .	331
ST_CoordDim . . . . .	332
ST_Crosses . . . . .	334
ST_Difference . . . . .	335
ST_Dimension . . . . .	337
ST_Disjoint . . . . .	338
ST_Distance . . . . .	340
ST_DistanceToPoint . . . . .	343
ST_Edge_GC_USA . . . . .	344
ST_Endpoint . . . . .	348
ST_Envelope . . . . .	349
ST_EnvIntersects . . . . .	350
ST_EqualCoordsys . . . . .	351
ST_Equals . . . . .	352
ST_EqualSRS . . . . .	354
ST_ExteriorRing . . . . .	355
ST_FindMeasure o ST_LocateAlong . . . . .	356
ST_Generalize . . . . .	357
ST_GeomCollection . . . . .	359
ST_GeomCollFromTxt . . . . .	361
ST_GeomCollFromWKB . . . . .	362
ST_Geometry . . . . .	364
ST_GeometryN . . . . .	365
ST_GeometryType . . . . .	367
ST_GeomFromText . . . . .	367
ST_GeomFromWKB . . . . .	369
ST_GetIndexParams . . . . .	370
ST_InteriorRingN . . . . .	373
ST_Intersection . . . . .	374
ST_Intersects . . . . .	375
ST_Is3d . . . . .	377
ST_IsClosed . . . . .	378
ST_IsEmpty . . . . .	380
ST_IsMeasured . . . . .	381
ST_IsRing . . . . .	382
ST_IsSimple . . . . .	383
ST_IsValid . . . . .	384
ST_Length . . . . .	385
ST_LineFromText . . . . .	387
ST_LineFromWKB . . . . .	388
ST_LineString . . . . .	389
ST_LineStringN . . . . .	390
ST_M . . . . .	392
ST_MaxM . . . . .	393
ST_MaxX . . . . .	394
ST_MaxY . . . . .	396
ST_MaxZ . . . . .	397
ST_MBR . . . . .	399
ST_MBRIntersects . . . . .	400
ST_MeasureBetween o ST_LocateBetween . . . . .	402
ST_MidPoint . . . . .	403
ST_MinM . . . . .	404
ST_MinX . . . . .	406

ST_MinY . . . . .	407
ST_MinZ . . . . .	408
ST_MLineFromText . . . . .	410
ST_MLineFromWKB . . . . .	411
ST_MPointFromText . . . . .	413
ST_MPointFromWKB . . . . .	414
ST_MPolyFromText . . . . .	415
ST_MPolyFromWKB . . . . .	416
ST_MultiLineString . . . . .	418
ST_MultiPoint . . . . .	420
ST_MultiPolygon . . . . .	421
ST_NumGeometries . . . . .	422
ST_NumInteriorRing . . . . .	423
ST_NumLineStrings . . . . .	424
ST_NumPoints . . . . .	425
ST_NumPolygons . . . . .	426
ST_Overlaps . . . . .	427
ST_Perimeter . . . . .	429
ST_PerpPoints . . . . .	431
ST_Point . . . . .	433
ST_PointAtDistance . . . . .	436
ST_PointFromText . . . . .	437
ST_PointFromWKB . . . . .	438
ST_PointN . . . . .	439
ST_PointOnSurface . . . . .	440
ST_PolyFromText . . . . .	441
ST_PolyFromWKB . . . . .	442
ST_Polygon . . . . .	443
ST_PolygonN . . . . .	445
ST_Relate . . . . .	446
ST_RemovePoint . . . . .	448
ST_SrsId, ST_SRID . . . . .	449
ST_SrsName . . . . .	450
ST_StartPoint . . . . .	451
ST_SymDifference . . . . .	452
ST_ToGeomColl . . . . .	454
ST_ToLineString . . . . .	456
ST_ToMultiLine . . . . .	457
ST_ToMultiPoint . . . . .	458
ST_ToMultiPolygon . . . . .	459
ST_ToPoint . . . . .	460
ST_ToPolygon . . . . .	461
ST_Touches . . . . .	462
ST_Transform . . . . .	463
ST_Union . . . . .	465
ST_Within . . . . .	467
ST_WKBToSQL . . . . .	469
ST_WKTToSQL . . . . .	470
ST_X . . . . .	471
ST_Y . . . . .	472
ST_Z . . . . .	473
Union aggregate (Agregado de unión) . . . . .	475

<b>Capítulo 24. Grupos de transformación. . . . .</b>	<b>477</b>
Grupos de transformación . . . . .	477
Grupo de transformación ST_WellKnownText . . . . .	477
Grupo de transformación ST_WellKnownBinary . . . . .	478
Grupo de transformación ST_Shape . . . . .	480
Grupo de transformación ST_GML . . . . .	481

**Capítulo 25. Formatos de datos soportados . . . . . 483**

Representación de texto convencional (well-known text, WKT) . . . . . 483  
Representación binaria convencional (well-known binary, WKB) . . . . . 488  
Representación de forma . . . . . 490  
Representación GML (Geography Markup Language) . . . . . 490

**Capítulo 26. Sistemas de coordenadas soportados . . . . . 491**

Sintaxis de los sistemas de coordenadas . . . . . 491  
    Unidades lineales soportadas . . . . . 493  
Unidades angulares soportadas . . . . . 493  
Esferoides soportados . . . . . 494  
Sistemas de referencia geodésicos soportados. . . 495  
Meridianos de origen soportados. . . . . 498  
Proyecciones cartográficas soportadas . . . . . 499

**Capítulo 27. Tareas espaciales desde el Centro de control de DB2 . . . . . 501**

Modificación de un sistema de coordenadas . . . 501  
Creación de un sistema de coordenadas . . . . . 501  
Creación de una columna espacial . . . . . 501  
Creación de un índice espacial . . . . . 502  
Ejecución de la geocodificación . . . . . 502

Definición de la geocodificación . . . . . 502  
Modificación de un sistema de referencia espacial . . . 503  
Importación de datos espaciales . . . . . 503

**Apéndice A. Visión general de la información técnica de DB2 . . . . . 505**

Biblioteca técnica de DB2 en copia impresa o en formato PDF . . . . . 505  
Pedido de manuales de DB2 en copia impresa . . . 508  
Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos. . . . . 509  
Acceso a diferentes versiones del Centro de información de DB2 . . . . . 509  
Visualización de temas en su idioma preferido en el Centro de información de DB2. . . . . 510  
Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet . 510  
Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet . . . . . 512  
Guías de aprendizaje de DB2 . . . . . 514  
Información de resolución de problemas de DB2 . . . 514  
Términos y condiciones . . . . . 515

**Apéndice B. Avisos . . . . . 517**

**Índice. . . . . 521**

---

# Capítulo 1. Acerca de DB2 Spatial Extender

Este capítulo proporciona información preliminar sobre DB2 Spatial Extender; describe su finalidad, los datos a los que da soporte y la forma en que están relacionados entre sí los conceptos subyacentes.

---

## La finalidad de DB2 Spatial Extender

Utilice DB2<sup>®</sup> Spatial Extender para generar y analizar información espacial sobre elementos geográficos y para almacenar y gestionar los datos en los que se basa esta información. Un elemento geográfico (algunas veces llamado elemento en esta explicación, para abreviar) es cualquier cosa del mundo real que tenga una ubicación identificable, o cualquier cosa que se pueda imaginar como existente en una ubicación identificable. Un elemento puede ser:

- Un objeto (es decir, una entidad concreta de cualquier tipo); por ejemplo, un río, un bosque o una cordillera de montañas.
- Un espacio; por ejemplo, una zona de seguridad alrededor de un lugar peligroso, o el área de mercado a la que da servicio un negocio en particular.
- Un suceso que se produce en una ubicación definible; por ejemplo, un accidente de circulación que se produce en un cruce de carreteras determinado, o una transacción de ventas en una tienda específica.

Los elementos existen en varios entornos. Por ejemplo, los objetos mencionados en la lista anterior: río, bosque, cordillera pertenecen al entorno natural. Otros objetos, tales como ciudades, edificios y oficinas, pertenecen al entorno cultural. Incluso otros, tales como parques, zoológicos y granjas, representan una combinación de los entornos natural y cultural.

En esta explicación, el término información espacial hace referencia al tipo de información que DB2 Spatial Extender pone a disposición de sus usuarios, es decir, hechos y cifras sobre las ubicaciones de elementos geográficos. Algunos ejemplos de información espacial son:

- Ubicaciones de elementos geográficos en el mapa (por ejemplo, los valores de longitud y latitud que definen dónde están situadas las ciudades)
- La ubicación de elementos geográficos con respecto a otros (por ejemplo, puntos dentro de una ciudad donde se encuentran hospitales y clínicas o la proximidad de las zonas residenciales de una ciudad con respecto a las zonas sísmicas locales)
- Modos en que los elementos geográficos se relacionan entre sí (por ejemplo, información sobre que un determinado sistema fluvial queda enclavado en una determinada región, o sobre que ciertos puentes de dicha región cruzan los afluentes del sistema fluvial)
- Las medidas que se utilizan para uno o varios elementos geográficos (por ejemplo, la distancia entre un edificio de oficinas y el límite del terreno o la longitud del perímetro de una reserva de aves)

La información espacial, ella sola o en combinación con datos relacionales tradicionales, puede ayudar al usuario en actividades como, por ejemplo, la definición de áreas en las que el usuario proporciona servicios, y en la determinación de ubicaciones de posibles mercados. Por ejemplo, supongamos que

el responsable de prestaciones sociales de un distrito tiene que comprobar cuáles de los candidatos y receptores de las prestaciones sociales viven realmente dentro del área a la que se aplican las prestaciones. DB2 Spatial Extender puede deducir esta información a partir de la ubicación del área en que se proporciona servicio y de las direcciones de los candidatos y de los receptores.

Supongamos ahora que el propietario de una cadena de restaurantes desea comenzar el negocio en ciudades cercanas. Para determinar dónde abrir nuevos restaurantes, el propietario necesita respuestas a preguntas como: ¿En qué puntos de estas ciudades se concentra la clientela que suele frecuentar mis restaurantes? ¿Dónde están las principales carreteras? ¿En qué puntos es más baja la tasa de criminalidad? ¿Dónde se encuentran los restaurantes de la competencia? DB2 Spatial Extender y DB2 pueden producir información para responder a estas preguntas. Además, las herramientas frontales, aunque no son necesarias, pueden contribuir. A modo de ilustración: una herramienta de visualización puede poner información generada por DB2 Spatial Extender (por ejemplo, la ubicación de las concentraciones de clientes y la proximidad de autopistas principales a los restaurantes propuestos) en forma geográfica en un mapa. Las herramientas de Business Intelligence pueden poner información asociada (por ejemplo, nombres y descripciones de restaurantes de la competencia) en forma de informe.

## Cómo los datos representan elementos geográficos

En DB2<sup>®</sup> Spatial Extender, se puede representar una característica geográfica mediante uno o más elementos de datos; por ejemplo, los elementos de datos de una tabla. (Un elemento de datos es el valor o los valores que ocupan una celda de una tabla relacional.) Por ejemplo, consideremos los edificios de oficinas y residencias. En la Figura 1, cada fila de la tabla BRANCHES representa una sucursal de un banco. De la misma forma, cada fila de la tabla CUSTOMERS de la Figura 1, considerada como una unidad, representa un cliente del banco. Sin embargo, un subconjunto de cada una de las filas (en concreto, de los elementos de datos que constituyen la dirección de un cliente) representa la residencia del cliente.

### BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA

### CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	95141	CA	USA	A	A

*Figura 1. Datos que representan elementos geográficos.* La fila de datos de la tabla BRANCHES representa una sucursal de un banco. Los datos de direcciones de la tabla CUSTOMERS representan la residencia de un cliente. Los nombres y las direcciones de ambas tablas son ficticios.

Las tablas de la Figura 1 contienen datos que identifican y describen las sucursales del banco y sus clientes. En esta explicación se hace referencia a datos de este tipo como datos comerciales.

Un subconjunto de los datos comerciales (los valores que representan las direcciones de las sucursales y de los clientes) se puede convertir en valores a partir de los cuales se genera la información espacial. Por ejemplo, tal como se

muestra en la Figura 1 en la página 2, la dirección de una sucursal es 92467 Airzone Blvd., San Jose, CA 95141, EE.UU. La dirección de un cliente es 9 Concourt Circle, San Jose, CA 95141, EE.UU. DB2 Spatial Extender puede convertir estas direcciones en valores que indiquen la dirección de la sucursal y la dirección privada del cliente, y dónde están situadas una respecto a la otra. La Figura 2 muestra las tablas BRANCHES y CUSTOMERS con nuevas columnas que están diseñadas para contener estos valores.

## BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA	

## CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	95141	CA	USA		A	A

Figura 2. Tablas con columnas espaciales añadidas. En cada tabla, la columna LOCATION contendrá coordenadas correspondientes a las direcciones.

Puesto que la información espacial se obtendrá de los datos almacenados en la columna LOCATION, en esta explicación se hace referencia a estos datos como datos espaciales.

## Naturaleza de los datos espaciales

Los datos espaciales están formados por coordenadas que identifican una ubicación. Spatial Extender trabaja con coordenadas bidimensionales especificadas por los valores de longitud y latitud  $x$  e  $y$ .

Una coordenada es un número que indica una de las dos cosas siguientes:

- Una posición a lo largo de un eje relativo a un origen, dada una unidad de longitud.
- Una dirección relativa a una línea o un plano básicos, dada una medida angular.

Por ejemplo, la latitud es una coordenada que indica un ángulo relativo al plano ecuatorial, normalmente en grados. La longitud es una coordenada que indica un ángulo relativo al meridiano de Greenwich y también se expresa normalmente en grados. De este modo, en un mapa, la posición del Parque Nacional de Yellowstone está definida por la latitud de 44,45 grados al norte del ecuador y la longitud de 110,40 grados al oeste del meridiano de Greenwich. Y más precisamente, estas coordenadas se refieren al centro del Parque Nacional de Yellowstone en Estados Unidos.

Las definiciones de latitud y longitud, sus puntos, líneas y planos de referencia, unidades de medida y otros parámetros asociados reciben en conjunto el nombre de sistema de coordenadas. Los sistemas de coordenadas pueden basarse en valores distintos de la latitud y la longitud. Estos sistemas de coordenadas tienen sus propios puntos, líneas y planos de referencia, unidades de medida y parámetros asociados adicionales (como la transformación de proyección).

Los datos espaciales más simples consisten de un único par de coordenadas que define la posición de una única ubicación geográfica. Un dato espacial más amplio consta de varias coordenadas que definen un recorrido lineal, tal como el que

formado por un camino o un río. Un tercer tipo consta de coordenadas que definen el perímetro de un área; por ejemplo, los límites de una parcela de tierra o de una zona con riesgo de inundación.

Cada dato espacial es un ejemplo de un tipo de dato espacial. El tipo de datos de las coordenadas que marcan una ubicación es `ST_Point`; el tipo de datos de las coordenadas que definen una vía de acceso lineal es `ST_LineString`, y el tipo de datos de las coordenadas que definen el perímetro de un área es `ST_Polygon`. Estos tipos, junto con los demás tipos de datos espaciales, son tipos estructurados que pertenecen a una jerarquía individual.

## Naturaleza de los datos geodésicos

Los datos geodésicos son datos espaciales que se expresan en coordenadas de latitud y longitud, en un sistema de coordenadas que describe una superficie cerrada, continua y redonda.

DB2 Geodetic Data Management Feature utiliza los mismos tipos de datos y funciones que Spatial Extender para almacenar datos geográficos en una base de datos DB2. A diferencia de Spatial Extender, que considera que la tierra es un mapa plano, Geodetic Data Management Feature la considera un globo sin bordes ni costuras en los polos o en las líneas de datos. Un mapa plano necesita coordenadas proyectadas para transformar las coordenadas esféricas en coordenadas planas. Sin embargo, Geodetic Data Management Feature utiliza la latitud y la longitud en un modelo elipsoidal de la superficie de la tierra. Los cálculos como la intersección lineal, el solapamiento de áreas, la distancia y el área son exactos y precisos, independientemente de su ubicación.

## De dónde proceden los datos espaciales

Puede obtener datos espaciales:

- A partir de datos comerciales
- A partir de funciones espaciales
- Importándolos de fuentes externas

### Utilización de datos comerciales como datos fuente

DB2 Spatial Extender puede obtener datos espaciales a partir de datos comerciales, tales como direcciones (tal como se menciona en el apartado “Cómo los datos representan elementos geográficos” en la página 2). Este proceso se denomina geocodificación. Para comprender la secuencia de procesos que intervienen, considere que la Figura 2 en la página 3 es una imagen “anterior” y la Figura 3 en la página 5 es una imagen “posterior”. La Figura 2 en la página 3 muestra que tanto la tabla `BRANCHES` como la tabla `CUSTOMERS` tienen una columna diseñada para datos espaciales. Supongamos que DB2 Spatial Extender geocodifica las direcciones de estas tablas para obtener coordenadas correspondientes a las direcciones y coloca las coordenadas en las columnas. La Figura 3 en la página 5 muestra este resultado.

## BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA	1653 3094

## CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourc Circle	San Jose	95141	CA	USA	953 1527	A	A

Figura 3. Tablas que incluyen datos espaciales obtenidos a partir de datos fuente. La columna LOCATION de la tabla CUSTOMERS contiene coordenadas que se han obtenido a partir de la dirección de las columnas ADDRESS, CITY, POSTAL CODE, STATE\_PROV y COUNTRY. De forma similar, la columna LOCATION de la tabla BRANCHES contiene coordenadas que se han obtenido a partir de la dirección de las columnas ADDRESS, CITY, POSTAL CODE, STATE\_PROV y COUNTRY de esta tabla.

DB2 Spatial Extender utiliza una función denominada geocodificador para convertir datos comerciales en coordenadas que permitan a las funciones espaciales trabajar con los datos.

### Utilización de funciones para generar datos espaciales

Puede utilizar funciones para generar datos espaciales a partir de los datos de entrada.

Los datos espaciales se pueden generar no solamente mediante geocodificadores, sino también mediante otras funciones. Por ejemplo, supongamos que el banco cuyas sucursales están definidas en la tabla BRANCHES desea saber el número de clientes situados a menos de cinco millas de cada sucursal. Antes de que el banco pueda obtener esta información de la base de datos, necesita definir la zona que queda dentro de un radio especificado alrededor de cada sucursal. Una función de DB2 Spatial Extender, ST\_Buffer, puede crear una definición de este tipo. Utilizando las coordenadas de cada sucursal como entrada, ST\_Buffer puede generar las coordenadas que delimitan los perímetros de las zonas. La Figura 4 muestra la tabla BRANCHES con información proporcionada por ST\_Buffer.

## BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION	SALES_AREA
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA	1653 3094	1002 2001, 1192 3564, 2502 3415, 1915 3394, 1002 2001

Figura 4. Tabla que incluye nuevos datos espaciales obtenidos a partir de datos espaciales existentes. Las coordenadas de la columna SALES\_AREA se han obtenido con la función ST\_Buffer a partir de las coordenadas de la columna LOCATION. Al igual que las coordenadas de la columna LOCATION, las de la columna SALES\_AREA son simuladas; no son reales.

Además de ST\_Buffer, DB2 Spatial Extender proporciona muchas otras funciones que generan nuevos datos espaciales a partir de datos espaciales existentes.

### Importación de datos espaciales

Spatial Extender proporciona servicios para importar datos espaciales en formato Shapefile.

Los datos espaciales en formato Shapefile están disponible desde muchas fuentes a través de Internet. Puede descargar datos y mapas para características de EE.UU. y

de todo el mundo tales como países, estados, ciudades, ríos, etc., seleccionando la oferta DB2 Spatial Extender Sample Map Data disponible en la página Web de pruebas y demostraciones, en <http://www.ibm.com/software/data/spatial/db2spatial>.

Puede importar datos espaciales suministrados por fuentes de datos externos. Estos archivos contienen habitualmente datos que se suelen aplicar a mapas: redes de calles, zonas con riesgo de inundación, fallas tectónicas, etcétera. La utilización de dichos datos en combinación con datos espaciales generados por el usuario, puede aumentar la información espacial disponible. Por ejemplo, si un departamento de obras públicas tuviera que determinar a qué peligros se expone una comunidad residencial, podría utilizar ST\_Buffer para definir una zona alrededor de la comunidad. El departamento de obras públicas podría entonces importar datos sobre zonas con riesgo de inundación y sobre fallas tectónicas para ver cuáles de estas áreas conflictivas quedan comprendidas dentro de esta zona.

---

## Cómo están interrelacionados los elementos geográficos, la información espacial, los datos espaciales y las geometrías

Esta sección resume varios conceptos básicos sobre los que están basadas las operaciones de DB2<sup>®</sup> Spatial Extender: los elementos geográficos, la información espacial y las geometrías.

DB2 Spatial Extender permite al usuario obtener datos y cifras referentes a objetos que se pueden definir geográficamente; es decir, en términos de su ubicación en la tierra o dentro de una región de la tierra. La documentación de DB2 hace referencia a datos y cifras tales como *información espacial* y objetos como *elementos geográficos* (llamados aquí simplemente *elementos* para abreviar).

Por ejemplo, podría utilizar DB2 Spatial Extender para determinar si una zona habitada queda dentro de una zona de vertidos propuesta. Las zonas habitadas y la zona de vertidos propuesta son elementos geográficos. El determinar si existe algún solapamiento entre ambas zonas sería un ejemplo de información espacial. Si se detecta que existe un solapamiento, su extensión sería también un ejemplo de información espacial.

Para producir información espacial, DB2 Spatial Extender debe procesar datos que definen las ubicaciones de los elementos. Estos datos, llamados *datos espaciales*, constan de coordenadas que indican las ubicaciones sobre una correlación o proyección similar. Por ejemplo, para determinar si un elemento geográfico se solapa con otro, DB2 Spatial Extender debe determinar dónde están situadas las coordenadas de uno de los elementos con respecto a las coordenadas del otro.

En la tecnología de la información espacial, es habitual pensar que los elementos geográficos se suelen representar mediante símbolos llamados *geometrías*. Las geometrías son entidades parcialmente visuales y parcialmente matemáticas. Considere su aspecto visual. El símbolo correspondiente a un elemento geográfico que tiene una anchura y extensión, tal como un parque o ciudad, es una figura de múltiples lados. Esta geometría se denomina *polígono*. El símbolo para un elemento lineal, tal como un río o una carretera, es una línea. Esta geometría se denomina *cadena lineal*.

La geometría representativa de un elemento geográfico tiene propiedades que se corresponden con datos referentes al elemento. La mayoría de estas propiedades se pueden expresar matemáticamente. Por ejemplo, las coordenadas de un elemento



geográfico forman colectivamente una de las propiedades de la geometría correspondiente del elemento. Otra propiedad, denominada *dimensión*, es un valor numérico que indica si un elemento geográfico tiene una longitud o extensión.

Los datos espaciales y determinada información espacial se pueden considerar en términos de geometrías. Considere el ejemplo, descrito anteriormente, de las zonas habitadas y la zona de vertidos propuesta. Los datos espaciales de las zonas habitadas comprenden coordenadas que están guardadas en una columna de una tabla de una base de datos de DB2. Por convenio, se considera que los datos almacenados no son simplemente datos, sino verdaderas geometrías. Debido a que las zonas habitadas tienen un ancho y una extensión, estas geometrías se pueden representar por polígonos.

Al igual que los datos espaciales, determinada información espacial se puede también considerar en términos de geometrías. Por ejemplo, para determinar si una zona habitada se solapa con una zona de vertidos propuesta, DB2 Spatial Extender debe comparar las coordenadas del polígono representativo del vertedero con las coordenadas de los polígonos representativos de las zonas habitadas. La información resultante (es decir, las zonas de solapamiento) se considera que son polígonos: geometrías con coordenadas, dimensiones y otras propiedades.



---

## Capítulo 2. Acerca de las geometrías

Este capítulo describe entidades de información, llamadas geometrías, que constan de coordenadas y representan elementos geográficos. Los temas que abarca son los siguientes:

- Geometrías
- Propiedades de las geometrías

---

### Geometrías

Según el diccionario revisado Webster de la lengua inglesa, la *geometría* es la “Rama de las matemáticas que investiga las relaciones, propiedades y mediciones de los cuerpos sólidos, superficies, líneas y ángulos; la ciencia que trata de las propiedades y relaciones de magnitudes; la ciencia de las relaciones del espacio.” La palabra geometría también se ha utilizado para denotar las funciones geométricas utilizadas durante más de un siglo por los cartógrafos para trazar mapas del mundo. Según esta nueva acepción, una definición abstracta de geometría sería: “punto o agregado de puntos que representa un elemento geográfico de la tierra.”

En DB2 Spatial Extender, la definición funcional de geometría es “un modelo de un elemento geográfico.” El modelo se puede expresar en términos de las coordenadas del elemento geográfico. El modelo expresa información: por ejemplo, las coordenadas identifican la posición del elemento geográfico con respecto a puntos fijos de referencia. Además, el modelo se puede utilizar para generar información; por ejemplo, la función ST\_Overlaps puede utilizar como datos de entrada las coordenadas de dos regiones próximas y notificar como información si las regiones se solapan o no.

Las coordenadas que un elemento geográfico representa se consideran propiedades de la geometría. Algunos tipos de geometrías tienen también otras propiedades; por ejemplo, área, longitud y perímetro.

Las geometrías soportadas por DB2 Spatial Extender forman una jerarquía, que se muestra en la figura siguiente. La jerarquía de la geometría está definida por el documento “OpenGIS Simple Features Specification for SQL” de OpenGIS Consortium, Inc. (OGC). Se pueden crear instancias para siete miembros de la jerarquía. Es decir, se pueden definir con valores de coordenadas específicos y representarse visualmente tal como se muestra en la figura.

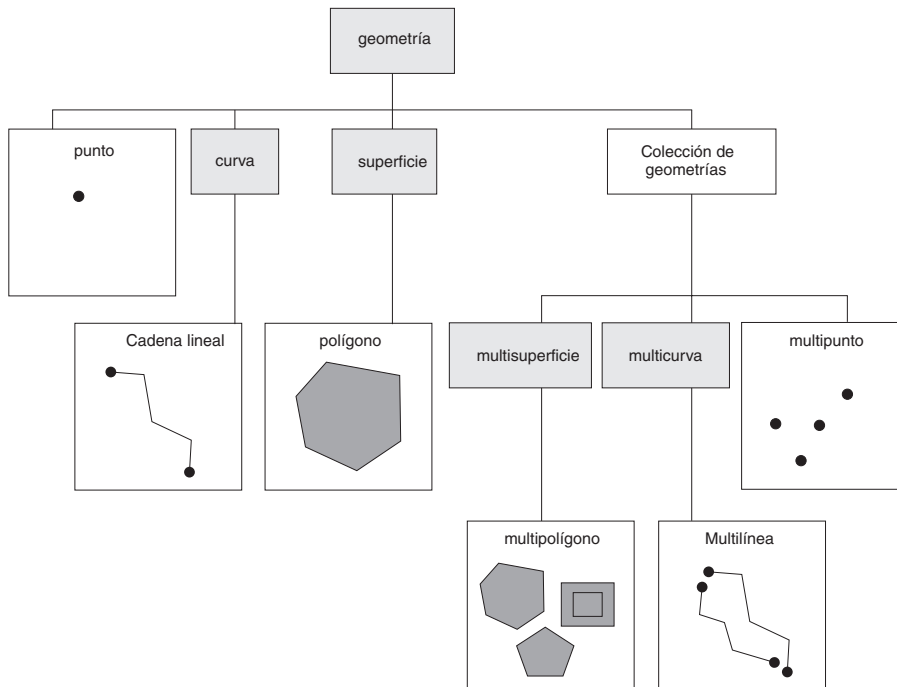


Figura 5. Jerarquía de geometrías soportadas por DB2 Spatial Extender. Las geometrías de esta figura de las que no pueden crearse instancias incluyen ejemplos de cómo se pueden representar visualmente.

Los tipos de datos espaciales soportados por DB2 Spatial Extender son implementaciones de las geometrías mostradas en la figura.

Tal como indica la figura, una superclase denominada geometría es la raíz de la jerarquía. Del tipo raíz y de otros subtipos adecuados de la jerarquía no pueden crearse instancias. Además, los usuarios pueden definir sus propios subtipos adecuados instanciables o no instanciables.

Los subtipos se dividen en dos categorías: los subtipos de geometrías base y los subtipos de colección homogénea.

Las geometrías base comprenden:

#### Puntos

Un único punto. Los puntos representan elementos discretos que se considera que ocupan el lugar donde se cruzan una línea de coordenadas este-oeste (como un paralelo) y una línea de coordenadas norte-sur (como un meridiano). Por ejemplo, suponga que la posición en un mapa del mundo muestra que cada ciudad del mapa está situada en la intersección de un paralelo y un meridiano. Un punto podría representar cada ciudad.

#### Cadenas lineales

Una línea entre dos o más puntos. No tiene por qué ser una línea recta. Las cadenas lineales representan elementos geográficos lineales (por ejemplo, calles, canales y conductos).

#### Polígonos

Un polígono o una superficie dentro de un polígono. Los polígonos representan elementos geográficos de múltiples lados (tales como barrios de viviendas sociales, bosques y hábitats salvajes).

Las colecciones homogéneas comprenden:

**Multipuntos**

Una colección de geometrías de puntos múltiples. Los multipuntos representan elementos geográficos que constan de varias partes cuyos componentes están situados en la intersección de una línea de coordenadas este-oeste y una línea de coordenadas norte-sur (por ejemplo, una cadena de islas cuyos miembros están situados en la intersección de un paralelo y un meridiano).

**Multilíneas**

Una colección de geometrías de curvas múltiples con múltiples cadenas lineales. Las multilíneas representan elementos geográficos que constan de varias partes (por ejemplo, sistemas de ríos y sistemas de carreteras).

**Multipolígonos**

Una colección de geometrías de superficies múltiples con múltiples polígonos. Los multipolígonos representan elementos que constan de varias partes y están formados por unidades con varios lados (por ejemplo, terrenos agrícolas colectivos de una región determinada o un sistema de lagos).

Tal como sus nombres indican, las colecciones homogéneas son colecciones de geometrías base. Además de tener las propiedades de las geometrías base, las colecciones homogéneas tienen también algunas propiedades propias.

---

## Propiedades de las geometrías

Este tema describe las propiedades de las geometrías. Estas propiedades son:

- El tipo al que pertenece la geometría
- Las coordenadas de la geometría
- El interior, perímetro y exterior de la geometría
- La cualidad de ser simple o no simple
- La cualidad de estar vacía o no vacía
- El rectángulo delimitador mínimo de la geometría, también llamado envoltura
- Dimensión
- Identificador de sistemas de referencia espacial que está asociado a esta geometría

### Tipo

Cada geometría pertenece a un tipo de la jerarquía de geometrías soportadas por DB2 Spatial Extender. Siete tipos de los puntos de la jerarquía (puntos, cadenas lineales, polígonos, colecciones de geometrías, multipuntos, multilíneas y multipolígonos) pueden definirse con valores de coordenadas específicas.

### Las coordenadas de la geometría

Todas las geometrías incluyen como mínimo una coordenada X y una coordenada Y, a menos que sean geometrías vacías, en cuyo caso no contendrán coordenadas de ningún tipo. Además, una geometría puede incluir una o más coordenadas Z y coordenadas M. Las coordenadas X, Y, Z y M se representan como números de precisión doble. En los siguientes subapartados se tratan estos temas:

- Coordenadas X e Y
- Coordenadas Z
- Coordenadas M

## Coordenadas X e Y

Un valor de coordenada X indica una posición con respecto a un punto de referencia situado al este u oeste. Un valor de coordenada Y indica una posición con respecto a un punto de referencia situado al norte o sur.

## Coordenadas Z

Algunas geometrías tienen asociado un valor de altitud o profundidad. Cada uno de los puntos que forman la geometría de un elemento geográfico puede incluir una coordenada Z opcional que representa una altitud o profundidad perpendicular a la superficie terrestre.

## Coordenadas M

Una coordenada M (medida) es un valor que expresa información sobre un elemento geográfico y que se guarda junto con las coordenadas que definen la posición del elemento geográfico. Por ejemplo, supongamos que está representando autopistas en su aplicación. Si desea que su aplicación procese valores que indican distancias lineales, puede guardar estos valores junto con las coordenadas que definen posiciones a lo largo de la autopista. Las coordenadas M se representan como números de precisión doble.

## Interior, perímetro y exterior

Todas las geometrías ocupan una posición en el espacio definido por sus interiores, perímetro y exteriores. El exterior de una geometría es todo el espacio no ocupado por la geometría. El perímetro de una geometría actúa de límite entre el interior y el exterior de la geometría. El interior es el espacio ocupado por la geometría.

## Simple o no simple

Los valores de algunos subtipos de geometría (cadenas lineales, multipuntos y multilíneas) pueden ser simples o no simples. Una geometría es simple si cumple todas las reglas topológicas impuestas sobre su subtipo, y es no simple en caso contrario. Una cadena lineal es simple si no forma intersección con su interior. Un multipunto es simple si ninguno de sus elementos ocupa el mismo espacio de coordenadas. Los puntos, superficies, multisuperficies y geometrías simples son siempre simples.

## Cerrada

Un curva es cerrada si los puntos inicial y final son los mismos. Una multicurva es cerrada si cada uno de sus elementos lo son. Un anillo es una curva simple y cerrada.

## Vacía o no vacía

Una geometría está vacía si no tiene ningún punto. La envoltura, el perímetro y el exterior de una geometría vacía no están definidos y se representarán como nulos. Una geometría vacía es siempre simple. Los polígonos y multipolígonos vacíos tienen un área igual a 0.

## Rectángulo delimitador mínimo (MBR)

El MBR de una geometría es la geometría delimitadora formada por las coordenadas (X,Y) mínimas y máximas. Excepto en los casos especiales siguientes, los MBR de las geometrías forman un rectángulo delimitador.

- El MBR de cualquier punto es el propio punto, pues sus coordenadas X máxima y mínima son iguales, y sus coordenadas Y máxima y mínima son iguales.

- El MBR de una cadena lineal horizontal o vertical es una cadena lineal representada por el perímetro (los puntos finales) de la cadena lineal original.

## Dimensión

Una geometría puede tener una dimensión de  $-1$ ,  $0$ ,  $1$  ó  $2$ . Las dimensiones aparecen listadas a continuación:

- $-1$  Está vacía
- $0$  Carece de longitud y su área es  $0$  (cero)
- $1$  Tiene una longitud mayor que  $0$  (cero) y su área es  $0$  (cero)
- $2$  Tiene un área mayor que  $0$  (cero)

Los subtipos punto y multipunto tienen una dimensión igual a cero. Los puntos representan elementos dimensionales que se pueden representar mediante un conjunto individual de coordenadas, mientras que los multipuntos denotan datos que se deben representar con un conjunto de puntos.

Los subtipos cadena lineal y multilínea tienen una dimensión igual a uno. Sirven para representar tramos de carretera, sistemas fluviales y otros elementos geográficos de carácter lineal.

Los subtipos polígono y multipolígono tienen una dimensión igual a 2. El tipo de datos de polígono y multipolígono puede representar elementos geográficos cuyo perímetro delimita un área definible, tales como bosques, parcelas de terreno y lagos.

## Identificador de sistemas de referencia espacial

El identificador de sistemas de referencia espacial determina qué sistema de referencia espacial se utiliza para representar una geometría.

Todos los sistemas de referencia espacial reconocidos por la base de datos se pueden acceder a través de la vista de catálogo `DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS`.





---

## Capítulo 3. Cómo utilizar DB2 Spatial Extender

---

### Cómo utilizar DB2 Spatial Extender

El soporte y la utilización de DB2<sup>®</sup> Spatial Extender implica dos actividades principales: configurar DB2 Spatial Extender y llevar a cabo proyectos que utilizan datos espaciales. Este tema presenta las interfaces que puede utilizar para realizar tareas espaciales.

#### Interfaces para DB2 Spatial Extender y funciones asociadas

Varias interfaces permiten al usuario configurar DB2 Spatial Extender y crear proyectos que utilizan datos espaciales. Estas interfaces son:

- Proyectos de fuente abierta que incluyen soporte para DB2 Spatial y Geodetic Data Management Features. Algunos proyectos fuente abiertos con este soporte incluyen:
  - GeoTools (<http://www.geotools.org/>), una biblioteca de Java<sup>™</sup> para construir aplicaciones espaciales
  - GeoServer (<http://docs.codehaus.org/display/GEOS/Home>), un servidor de mapas web y un servidor de características web
  - uDIG (<http://udig.refractor.net/confluence/display/UDIG/Home>), una aplicación de visualización y análisis de datos espaciales de escritorio
- El Centro de control de DB2, una interfaz gráfica de usuario que incluye ventanas, cuadernos y selecciones de menús que dan soporte a DB2 Spatial Extender.
- Un procesador de línea de mandatos (CLP) proporcionado por DB2 Spatial Extender. Se denomina CLP de db2se.
- Programas de aplicación que llaman a procedimientos almacenados de DB2 Spatial Extender.

Otras interfaces permiten al usuario generar información espacial. Entre ellas se incluyen:

- Consultas SQL que el usuario envía desde el CLP de DB2, desde una ventana de consulta del Centro de control de DB2 o desde un programa de aplicación.
- Herramientas de visualización que producen información espacial en forma gráfica. Un ejemplo es ArcExplorer para DB2, que el instituto Environmental Systems Research Institute (ESRI) creó para IBM<sup>®</sup>. ArcExplorer para DB2 se puede bajar desde el sitio Web de DB2 Spatial Extender: <http://www.ibm.com/software/data/spatial/>.

#### Tareas que el usuario realiza para configurar DB2 Spatial Extender y crear proyectos

Esta sección proporciona una visión general de las tareas que el usuario tiene que realizar para configurar DB2 Spatial Extender y trabajar en proyectos que utilicen datos espaciales. Se incluye un caso práctico que ilustra las tareas. Las tareas se dividen en dos categorías:

- Configuración de DB2 Spatial Extender
- Creación de proyectos que utilizan datos espaciales

## Configuración de DB2 Spatial Extender

Esta sección enumera las tareas que el usuario puede realizar para configurar DB2 Spatial Extender y utiliza un caso práctico para ilustrar cómo una empresa ficticia puede afrontar cada tarea.

### Para configurar DB2 Spatial Extender:

1. Planifique y lleve a cabo la preparación (decida qué proyectos crear, decida qué interfaz o interfaces utilizar, seleccione personal para administrar DB2 Spatial Extender y crear los proyectos, etcétera).

**Caso práctico:** El entorno de sistemas de información de la compañía de seguros de inmuebles Safe Harbor Real Estate incluye un sistema de base de datos DB2 y un sistema de archivos separado para datos espaciales únicamente. Como extensión, los resultados de consulta pueden incluir combinaciones de datos de ambos sistemas. Por ejemplo, una tabla de DB2 almacena información sobre ingresos, y un archivo del sistema de archivos contiene las ubicaciones de las sucursales de la compañía. Por lo tanto, es posible averiguar qué oficinas aportan ingresos de cantidades especificadas y, a continuación, determinar dónde están ubicadas esas oficinas. Pero los datos procedentes de los dos sistemas no se pueden integrar (por ejemplo, los usuarios no pueden unir columnas de DB2 con registros del sistema de archivos, y los servicios de DB2, tales como la optimización de consultas, no están disponibles para el sistema de archivos.) Para solucionar estos problemas, Safe Harbor adquiere DB2 Spatial Extender y establece un nuevo departamento de desarrollo espacial (llamado Departamento espacial, en su forma abreviada).

La primera misión del Departamento espacial consiste en incluir DB2 Spatial Extender en el entorno de DB2 de Safe Harbor:

- El equipo de dirección del departamento asigna a un equipo de administración espacial la tarea de instalar e implantar DB2 Spatial Extender, y a un equipo de análisis espacial la tarea de generar y analizar información espacial.
  - Puesto que el equipo de administración tiene una sólida formación en UNIX®, decide utilizar el CLP de db2se para administrar DB2 Spatial Extender.
  - Puesto que las decisiones comerciales de Safe Harbor se basan principalmente en los requisitos de los clientes, el equipo de dirección decide instalar DB2 Spatial Extender en la base de datos que contiene información sobre los clientes. La mayor parte de esta información se almacena en una tabla denominada CUSTOMERS.
2. Instale DB2 Spatial Extender.  
**Caso práctico:** El equipo de administración espacial instala DB2 Spatial Extender en una máquina UNIX® dentro de un entorno DB2.
  3. Si dispone de DB2 Spatial Extender Versión 8, migre los datos espaciales a DB2 Versión 9.5.  
**Caso práctico:** La versión 9.5 es la primera que Safe Harbor adquiere. No es necesaria ninguna migración.
  4. Configure la base de datos para alojar los datos espaciales. Ajuste los parámetros de configuración para asegurar que la base de datos tenga memoria y espacio suficiente para funciones espaciales, archivos de anotaciones cronológicas y aplicaciones de DB2 Spatial Extender.

**Caso práctico:** Un miembro del equipo de administración espacial ajusta las características de los archivos de anotaciones cronológicas de transacción, el

tamaño de la pila de aplicaciones y el tamaño de pila de control de aplicaciones de acuerdo con los valores adecuados para los requisitos de DB2 Spatial Extender.

5. Configure los recursos espaciales para la base de datos. Estos recursos incluyen un catálogo del sistema, tipos de datos espaciales, funciones espaciales, un geocodificador y otros objetos. La tarea de configurar estos recursos se denomina habilitación de la base de datos para operaciones espaciales.

El geocodificador proporcionado por DB2 Spatial Extender convierte direcciones de Estados Unidos en datos espaciales. Se denomina DB2SE\_USA\_GEOCODER. Su organización y otras pueden proporcionar geocodificadores que conviertan direcciones dentro o fuera de los estados unidos así como otros tipos de datos en datos espaciales.

**Caso práctico:** El equipo de administración espacial prepara los recursos que serán necesarios para los proyectos que planean.

- Un miembro del equipo emite un mandato para obtener los recursos que habilitan la base de datos para operaciones espaciales. Estos recursos incluyen el catálogo de DB2 Spatial Extender, tipos de datos espaciales, funciones espaciales, etcétera.
- Puesto que Safe Harbor está empezando a realizar operaciones comerciales en Canadá, el equipo de administración espacial empieza solicitando a los proveedores canadienses geocodificadores para convertir las direcciones de Canadá en datos espaciales. Safe Harbor no tiene intención de adquirir tales geocodificadores al menos durante unos meses. Por lo tanto, las primeras ubicaciones en las que reunirá datos serán en Estados Unidos.

## Creación de proyectos que utilizan datos espaciales

Después de configurar DB2 Spatial Extender, está preparado para emprender proyectos que utilicen datos espaciales. Esta sección enumera las tareas implicadas en la creación de un proyecto y continúa el caso práctico en el que la empresa de seguros de inmuebles Safe Harbor busca integrar datos comerciales y datos espaciales.

### Para crear un proyecto que utilice datos espaciales:

1. Planifique y lleve a cabo la preparación (defina objetivos del proyecto, decida qué tablas y datos necesita, determine qué sistema o sistemas de coordenadas utilizar, etc.)

**Caso práctico:** El departamento se prepara para desarrollar el proyecto; por ejemplo:

- El equipo de dirección establece estos objetivos para el proyecto:
  - Determinar dónde establecer nuevas sucursales
  - Fijar primas de riesgo en función de la proximidad de los clientes a zonas de riesgo (zonas con tasas altas de accidentes de tráfico, zonas con un nivel alto de criminalidad, zonas con riesgo de inundación, fallas tectónicas, etcétera.)
- Este proyecto particular tratará con clientes y oficinas de Estados Unidos. Por lo tanto, el equipo de administración espacial decide:
  - Utilizar un sistema de coordenadas para Estados Unidos que DB2 Spatial Extender suministra. Se denomina GCS\_NORTH\_AMERICAN\_1983.
  - Utilizar el geocodificador DB2SE\_USA\_GEOCODER, porque está diseñado para geocodificar direcciones de Estados Unidos.

- El equipo de administración espacial decide qué datos se necesitan para cumplir con los objetivos del proyecto y qué tablas contendrán dichos datos.
2. Cree un sistema de coordenadas si el usuario necesita hacerlo así.

**Caso práctico:** Porque Safe Harbor ha decidido utilizar GCS\_NORTH\_AMERICAN\_1983, la compañía puede omitir este paso.

3. Decida si un sistema de referencia espacial existente se ajusta a sus necesidades. Si ninguno lo hace, cree uno.

Un sistema de referencia espacial es un conjunto de valores de parámetros que incluye:

- Coordenadas que definen la máxima extensión de espacio posible al que se hace referencia mediante un rango determinado de coordenadas. El usuario necesita determinar el máximo rango de coordenadas posible que se pueda determinar desde el sistema de coordenadas que esté utilizando, y seleccionar o crear un sistema de referencia espacial que refleje este rango.
- El nombre del sistema de coordenadas del cual se obtienen las coordenadas.
- Los números utilizados en operaciones matemáticas para convertir coordenadas recibidas como datos de entrada en valores que se puedan procesar con máxima eficacia. Las coordenadas se guardan en su formato convertido y se devuelven al usuario en su formato original.

**Caso práctico:** DB2 Spatial Extender proporciona un sistema de referencia espacial, NAD83\_SRS\_1, que está diseñado para su utilización con GCS\_NORTH\_AMERICAN\_1983. El equipo de administración espacial decide utilizar NAD83\_SRS\_1.

4. Cree columnas espaciales, según sea necesario. Se ha de tener en cuenta que en muchos casos, si una herramienta de visualización va a leer los datos de una columna espacial, la columna debe ser la única columna espacial de la tabla o vista a la que pertenece. De forma alternativa, si la columna es una de varias columnas espaciales en una tabla, se podría incluir en una vista que no tenga otras columnas espaciales, y las herramientas de visualización podrían leer los datos de esta vista.

**Caso práctico:** El equipo de administración espacial define columnas para que contengan datos espaciales.

- El equipo añade una columna LOCATION a la tabla CUSTOMERS. La tabla ya contiene las direcciones de los clientes. DB2SE\_USA\_GEOCODER las convertirá en datos espaciales. A continuación, DB2 almacenará estos datos en la columna LOCATION.
  - El equipo crea una tabla OFFICE\_LOCATIONS y una tabla OFFICE\_SALES para que contengan los datos que ahora se almacenan en el sistema de archivos separado. Estos datos incluyen las direcciones de las sucursales de Safe Harbor, datos espaciales que el geocodificador ha obtenido a partir de estas direcciones y datos espaciales que definen una zona dentro de un radio de cinco millas alrededor de cada sucursal. Los datos obtenidos por el geocodificador irán dentro de una columna LOCATION de la tabla OFFICE\_LOCATIONS, y los datos que definen las zonas irán dentro de una columna SALES\_AREA de la tabla OFFICE\_SALES.
5. Defina columnas espaciales para el acceso mediante herramientas de visualización, según sea necesario. El usuario realiza esta acción registrando las columnas en el catálogo de DB2 Spatial Extender. Cuando se registra una columna espacial, DB2 Spatial Extender impone una restricción de que todos los datos de la columna deben pertenecer al mismo sistema de referencia espacial. Esta restricción conlleva la integridad de los datos: un requisito de la mayoría de herramientas de visualización.

**Caso práctico:** El equipo de administración espacial espera utilizar herramientas de visualización para producir el contenido de las columnas LOCATION y de la columna SALES\_AREA gráficamente en un mapa. Por lo tanto, el equipo registra las tres columnas.

6. Llene con datos columnas espaciales:

Para un proyecto que necesite que se importen datos espaciales, importe los datos.

Para un proyecto que necesite un geocodificador:

- Establezca, por adelantado, la información de control necesaria cuando se invoca un geocodificador.
- Como opción, configure el geocodificador para que se ejecute automáticamente cada vez que se añada una nueva dirección a la base de datos o que se actualice una dirección existente.

Ejecute el geocodificador en modalidad de proceso por lotes, según sea necesario.

Para un proyecto que requiera que se creen datos espaciales mediante una función espacial, ejecute esta función.

**Caso práctico:** El equipo de administración espacial llena la columna LOCATION de la tabla CUSTOMER, la tabla OFFICE\_LOCATIONS, la tabla OFFICE\_SALES y una nueva tabla HAZARD\_ZONES:

- El equipo utiliza DB2SE\_USA\_GEOCODER para geocodificar direcciones de la tabla CUSTOMER. Las coordenadas producidas por la geocodificación se insertan en la columna LOCATION de la tabla.
- El equipo utiliza un programa de utilidad para cargar datos de oficina desde el sistema de archivos en un archivo. A continuación, el equipo importa estos datos a la nueva tabla OFFICE\_LOCATIONS.
- El equipo crea una tabla HAZARD\_ZONES, registra sus columnas espaciales e importa los datos a la misma. Los datos proceden de un archivo suministrado por un proveedor de mapas.

7. Facilite el acceso a columnas espaciales, según sea necesario. Eso implica la definición de índices que posibiliten a DB2 acceder a datos espaciales de forma rápida y la definición de vistas que permitan a los usuarios recuperar eficazmente datos interrelacionados. Si se desea que las herramientas de visualización accedan a las columnas espaciales de la vista, es necesario registrar estas columnas también con DB2 Spatial Extender.

**Caso práctico:** El equipo de administración espacial crea índices para las columnas registradas. A continuación, el equipo crea una vista que une columnas procedentes de las tablas CUSTOMERS y HAZARD\_ZONES. A continuación, registra las columnas espaciales de la vista.

8. Genere información espacial e información comercial asociada. Analice la información. Esta tarea implica consultar columnas espaciales y columnas no espaciales asociadas. En tales consultas, puede incluir funciones de DB2 Spatial Extender que devuelven una amplia variedad de información; por ejemplo, coordenadas que definen una zona de seguridad propuesta alrededor de una zona de vertidos peligrosa, o la distancia mínima entre el vertedero y el edificio público más cercano.

**Caso práctico:** El equipo de análisis espacial ejecuta consultas para obtener información que le ayudará a cumplir con los objetivos originales: determinar dónde establecer nuevas sucursales y ajustar las primas en función de la proximidad de los clientes a zonas de riesgo.



---

## Capítulo 4. Iniciación a DB2 Spatial Extender

Este capítulo proporciona instrucciones para instalar y configurar Spatial Extender para los entornos operativos AIX, HP-UX, Windows®, Linux®, Linux en IBM System z y Solaris. Este capítulo también explica cómo resolver algunos problemas de instalación y configuración que pueden producirse al invocar Spatial Extender.

---

### Configuración e instalación de Spatial Extender

#### Requisitos previos

Antes de instalar DB2 Spatial Extender, debe tener un servidor de datos y un cliente DB2 instalados en un solo sistema o un servidor de datos DB2 instalado en un sistema y un cliente DB2 instalado en otro sistema.

Un entorno DB2 Spatial Extender consta de una instalación de servidor de datos DB2 y una instalación de DB2 Spatial Extender. Las bases de datos habilitadas para las operaciones espaciales se encuentran en el servidor de datos DB2, al que puede accederse desde un cliente DB2 Spatial Extender. Un servidor de datos DB2 y un cliente DB2 Spatial Extender pueden instalarse en el mismo sistema. Se puede acceder a los datos espaciales que residen en las bases de datos utilizando los procedimientos almacenados y las consultas espaciales de DB2 Spatial Extender. DB2 Geodetic Data Management Feature se incluye con DB2 Spatial Extender, no obstante, se necesita otra licencia para utilizarlo. También suele formar parte de un sistema DB2 Spatial Extender típico un geonavegador que, aunque no es necesario, resulta útil para presentar visualmente los resultados de las consultas espaciales, generalmente en forma de mapas. Una instalación DB2 Spatial Extender no incluye un geonavegador, no obstante, se pueden ver datos espaciales con geonavegadores como el geonavegador de fuente abierta, ArcExplorer para DB2 o los conjuntos de herramientas ArcGIS de ESRI que se ejecutan con ArcSDE.

Para realizar esta tarea:

1. Asegúrese de que el sistema cumple con todos los requisitos de software. Consulte la sección “Requisitos del sistema para instalar Spatial Extender” en el manual *Spatial Extender and Geodetic Data Management Feature Guía del usuario y manual de consulta*
2. Instale Spatial Extender. Si el sistema no cumple ninguno de los requisitos previos de software, no se realizará la instalación. Consulte la sección “Instalación de DB2 Spatial Extender (Windows)” en la página 22 o “Instalación de DB2 Spatial Extender (Linux, UNIX)” en la página 23
3. Cree una instancia de DB2 si todavía no tiene ninguna. Para ello, utilice el mandato db2icrt DB2 desde una ventana de mandatos de DB2.
4. Verifique que la instalación de Spatial Extender se ha realizado correctamente probando el entorno de Spatial Extender. Consulte la sección “Verificación de la instalación de Spatial Extender” en la página 24
5. OPCIONAL: Descargue e instale un geonavegador como ArcExplorer para DB2 o los conjuntos de herramientas ArcGIS de ESRI que se ejecutan con ArcSDE. Se puede descargar una copia gratuita de ArcExplorer para DB2 desde el sitio Web de IBM DB2 Spatial Extender: <http://www.ibm.com/software/data/spatial/db2spatial/>

---

## Requisitos del sistema para instalar Spatial Extender

Antes de instalar DB2 Spatial Extender, asegúrese de que el sistema cumple todos los requisitos de software y espacio de disco descritos abajo.

### Sistemas operativos

Puede instalar DB2 Spatial Extender en sistemas operativos de 32 bits como Windows o Linux en sistemas basados en Intel. También puede instalar DB2 Spatial Extender en sistemas operativos de 64 bits como AIX, HP-UX PA-RISC, Sistema Operativo Solaris SPARC, Linux x86, Linux para System z y Windows.

### Requisitos de software

Para instalar Spatial Extender, debe tener el siguiente software de DB2 instalado y configurado:

#### Software del servidor

Debe instalar DB2 *antes* de instalar DB2 Spatial Extender.

Puede utilizar la interfaz gráfica de usuario del Centro de control de DB2 para realizar algunas tareas de Spatial Extender. Si desea utilizar esta interfaz, cree y configure el Servidor de administración de DB2 (DAS).

Para obtener más información sobre el soporte de Spatial Extender del Centro de control, consulte el tema Capítulo 27, “Tareas espaciales desde el Centro de control de DB2”, en la página 501

Para obtener más información sobre cómo crear y configurar DAS, consulte el tema: [.././com.ibm.db2.luw.qb.server.doc/doc/t0006743.dita](#)

#### Software del cliente espacial

Si instala DB2 Spatial Extender en Windows, la instalación por omisión de Spatial Extender incluye el cliente espacial. Cuando instale DB2 Spatial Extender en sistemas operativos AIX, HP-UX, Solaris, Linux para Intel® o Linux en System z, puede instalar opcionalmente el cliente espacial al instalar un servidor de datos y un cliente DB2.

### Requisitos de espacio en disco

Para instalar Spatial Extender, el sistema debe cumplir los requisitos de espacio en disco identificados durante el proceso de instalación, de lo contrario, la instalación fallará y aparecerá un mensaje de error que indique que el espacio en disco disponible no es suficiente.

---

## Instalación de DB2 Spatial Extender (Windows)

La instalación de DB2 Spatial Extender en sistemas operativos Windows requiere que la tarea siguiente se complete correctamente.

Antes de instalar la característica DB2 Spatial Extender, debe instalarse un producto de servidor de datos DB2.

Esta tarea forma parte de la tarea más amplia de “Configuración e instalación de Spatial Extender” en la página 21



Puede instalar DB2 Spatial Extender en sistemas operativos Windows utilizando el Asistente de instalación de DB2 o un archivo de respuestas.

**Recomendación:**

Utilice el asistente de instalación de DB2 para instalar Spatial Extender. El asistente de instalación proporciona una interfaz gráfica fácil de utilizar que puede utilizarse para crear instancias, automatizar la creación de usuarios y grupos, instalar configuraciones de protocolo y acceder a la ayuda de instalación.

Si está utilizando el Asistente de instalación de DB2 para instalar Spatial Extender, puede pulsar **Cancelar** en cualquier momento de la instalación para salir del proceso.

En cualquier momento durante la instalación puede pulsar Ayuda para iniciar la ayuda de instalación en línea.

## Instalación de Spatial Extender mediante el asistente de instalación

1. Inicie la sesión en el sistema con la cuenta de usuario que desea utilizar para realizar la instalación.
2. Inserte y monte el CD o DVD de Spatial Extender en la unidad de CD o DVD. Si el programa de instalación no se abre automáticamente, ejecútelo emitiendo el mandato `setup.exe` si no se ejecuta automáticamente.
3. Ejecute el programa de instalación emitiendo el mandato `setup` desde un indicador de mandatos.
4. Una vez finalizada la instalación, compruebe si hay mensajes de aviso o error en el archivo de anotaciones cronológicas identificado.

La instalación debe completarse correctamente. Si se han producido errores durante el proceso de instalación, la instalación se detendrá antes de finalizar.

## Instalación de Spatial Extender mediante un archivo de respuestas

Consulte la sección `../../com.ibm.db2.luw.qb.server.doc/doc/t0007313.dita`

---

## Instalación de DB2 Spatial Extender (Linux, UNIX)

La instalación de DB2 Spatial Extender en sistemas operativos Linux o UNIX requiere que la tarea siguiente se complete correctamente.

Requisitos previos

Antes de instalar la característica DB2 Spatial Extender, debe instalarse un producto de servidor de datos DB2.

Esta tarea forma parte de la tarea más amplia de “Configuración e instalación de Spatial Extender” en la página 21

Puede instalar DB2 Spatial Extender en sistemas operativos Linux y UNIX utilizando el asistente de instalación de DB2, el mandato `db2_install` o un archivo de respuestas.

**Recomendación:** Utilice el asistente de instalación de DB2 para instalar Spatial Extender. El asistente de instalación proporciona una interfaz gráfica fácil de utilizar que puede utilizarse para crear instancias, automatizar la creación de usuarios y grupos, instalar configuraciones de protocolo y acceder a la ayuda de instalación.

Si está utilizando el Asistente de instalación de DB2 para instalar Spatial Extender, puede pulsar **Cancelar** en cualquier momento de la instalación para salir del proceso.

En cualquier momento durante la instalación puede pulsar Ayuda para iniciar la ayuda de instalación en línea.

Después de instalar Spatial Extender, cree el entorno de instancias de DB2 si no lo ha hecho aún y, a continuación, verifique la instalación.

## Instalación de DB2 Spatial Extender utilizando el asistente de instalación de DB2 (Linux, UNIX)

1. Inserte y monte el CD o DVD de Spatial Extender en la unidad de CD o DVD del sistema cliente. Se abre el Área de ejecución para la instalación de DB2, una interfaz desde la que puede instalar DB2 Spatial Extender.
2. Seleccione DB2 Spatial Extender como el producto que desea instalar y pulse **SIGUIENTE**.
3. Pulse **Instalar un producto**.
4. Pulse el botón **Trabajar con existente**. Seleccione una copia existente de un servidor de datos DB2 en el que instalar Spatial Extender.
5. Pulse **Iniciar el Asistente de instalación de DB2**. Se abre la ventana Asistente de instalación de DB2. Utilice el asistente de instalación de DB2 para que le guíe en los pasos restantes de instalación y configuración.

La instalación debe completarse correctamente. Si se han producido errores durante el proceso de instalación, la instalación se detendrá antes de finalizar.

## Instalación de Spatial Extender mediante el mandato db2\_install (Linux, UNIX)

1. Inserte y monte el CD apropiado.
2. Escriba el mandato `./db2_install` para iniciar el script `db2_install`. El script `db2_install` se encuentra en el directorio raíz del CD del producto DB2. El script `db2_install` le solicita la palabra clave del producto.
3. Escriba **GSE** para instalar DB2 Spatial Extender.

## Instalación de Spatial Extender mediante un archivo de respuestas

Consulte la sección `../../com.ibm.db2.luw.qb.server.doc/doc/t0007312.dita`

---

## Verificación de la instalación de Spatial Extender

Después de instalar DB2 Spatial Extender, es recomendable validar la instalación.

### Antes de empezar

Antes de validar una instalación de Spatial Extender deben cumplirse los siguientes requisitos previos:

- Spatial Extender debe estar instalado en un sistema.
- Deberá haberse creado una instancia de DB2 en el sistema donde está instalado el servidor de datos DB2.

### Acerca de esta tarea

Esta tarea debe realizarse después de instalar y configurar Spatial Extender. Consulte la sección “Configuración e instalación de Spatial Extender” en la página 21. La tarea consiste en crear una base de datos DB2 y ejecutar una aplicación de ejemplo de Spatial Extender que se suministra con DB2 y que puede utilizarse para verificar que un conjunto principal de funciones de DB2 Spatial Extender funciona correctamente. Esta prueba es suficiente para validar que DB2 Spatial Extender se ha instalado y configurado correctamente.

### Procedimiento

Para realizar esta tarea:

1. Linux y UNIX: Inicie sesión en el sistema con el ID de usuario que corresponda al rol de propietario de la instancia de DB2.
2. Cree una base de datos DB2. Para ello, abra una ventana de mandatos de DB2 y especifique lo siguiente:

```
db2 create database mi_bd
```

donde *mi\_bd* es el nombre de la base de datos.

3. Asegúrese de que dispone de un espacio de tablas temporal del sistema con un tamaño de página de 8 KB o más y un tamaño mínimo de 500 páginas. Si no dispone de un espacio de tablas con estas características, consulte la sección sobre “creación de espacios de tablas temporales” en *Database Administration Concepts and Configuration Reference* para obtener más información sobre cómo crearlo. Es un requisito para ejecutar el programa *runGSEdemo*.
4. Aumente el tamaño de la pila de aplicaciones de la base de datos DB2 para que la base de datos pueda acomodar los requisitos de espacio más amplio de los datos espaciales. Para obtener más información, consulte la sección “Configuración de una base de datos para alojar datos espaciales” en la página 29.
5. Vaya al directorio donde reside el programa *runGSEdemo*:

- Para las instalaciones de Spatial Extender en sistemas operativos Linux y UNIX, especifique lo siguiente:

```
cd $HOME/sqlllib/samples/extenders/spatial
```

donde *\$HOME* es el directorio inicial del propietario de la instancia.

- Para la instalación de Spatial Extender en sistemas operativos Windows, especifique lo siguiente:

```
cd c:\Archivos de programa\IBM\sqlllib\samples\extenders\spatial
```

donde *c:\Archivos de programa\IBM\sqlllib* es el directorio en el que ha instalado DB2 Spatial Extender.

6. Ejecute el programa de verificación de la instalación. En la línea de mandatos de DB2, escriba el mandato *runGseDemo*:

```
runGseDemo mi_bd ID_usuario contraseña
```

donde *mi\_bd* es el nombre de la base de datos.

---

## Consideraciones posteriores a la instalación

Después de instalar Spatial Extender, tenga en cuenta las acciones siguientes:

- OPCIONAL: Descargue e instale un geonavegador como ArcExplorer para DB2 o los conjuntos de herramientas ArcGIS de ESRI que se ejecutan con ArcSDE. Se puede descargar una copia gratuita de ArcExplorer para DB2 desde el sitio Web de IBM DB2 Spatial Extender: <http://www.ibm.com/software/data/spatial/db2spatial/>

## Descarga de ArcExplorer para DB2

IBM proporciona un navegador, creado por ESRI (Environmental Systems Research Institute) para IBM, que puede producir directamente resultados visuales para consultas de datos de DB2 Spatial Extender sin necesidad de un servidor de datos intermedio. Este navegador es ArcExplorer para DB2. Puede descargar una copia gratuita de ArcExplorer para DB2 desde el sitio web de Spatial Extender de IBM en la dirección siguiente:

<http://www.ibm.com/software/data/spatial/db2spatial/>

Para obtener más información sobre cómo instalar y utilizar ArcExplorer para DB2, consulte *Utilización de ArcExplorer*, que también está disponible como parte del producto ArcExplorer para DB2 que puede bajar en el sitio web de DB2 Spatial Extender.

**Importante:** Instale ArcExplorer para DB2, en un directorio que esté separado de DB2.

---

## Capítulo 5. Actualización a DB2 Spatial Extender Versión 9.7

Actualizar DB2 Spatial Extender a la versión 9.7 en sistemas en los que ha instalado DB2 Spatial Extender Versión 8, Versión 9.1 o Versión 9.5 requiere más que instalar DB2 Spatial Extender versión 9.7. Debe realizar la tarea de actualización adecuada para estos sistemas.

En las tareas siguientes se describen todos los pasos para actualizar DB2 Spatial Extender de la versión 8, versión 9.1 o versión 9.5 a la versión 9.7:

- “Actualización de DB2 Spatial Extender”
- “Actualización de DB2 Spatial Extender” en la página 28

Si su entorno DB2 tiene otros componentes, como servidores de DB2, clientes y aplicaciones de base de datos, consulte “Actualización a DB2 Versión 9.7” en *Actualización a DB2 Versión 9.7* para obtener más detalles sobre cómo actualizar estos componentes.

---

### Actualización de DB2 Spatial Extender

La actualización de DB2 Spatial Extender requiere que el usuario actualice primero el servidor DB2 y, a continuación, actualice los objetos de base de datos específicos y los datos de bases de datos espaciales.

#### Antes de empezar

Antes de empezar el proceso de actualización:

- Asegúrese de que el sistema cumple los requisitos de instalación de DB2 Spatial Extender Versión 9.7.
- Asegúrese de que tiene autorización DBADM y DATAACCESS en las bases de datos habilitadas para funciones espaciales.
- Asegúrese de que dispone de un espacio de tablas temporal del sistema con un tamaño de página de 8 KB o más y un tamaño mínimo de 500 páginas. Si no dispone de un espacio de tablas con estas características, consulte la sección sobre “creación de espacios de tablas temporales” en *Database Administration Concepts and Configuration Reference* para obtener más información sobre cómo crearlo.

#### Acerca de esta tarea

Si ha instalado DB2 Spatial Extender Versión 8, Versión 9.1 o Versión 9.5, debe completar los pasos siguientes antes de utilizar una base de datos existente habilitada para operaciones espaciales con DB2 Spatial Extender Versión 9.7 o DB2 Geodetic Data Management Feature Versión 9.7. Este tema describe los pasos necesarios que hay que llevar a cabo para actualizar bases de datos habilitadas para operaciones espaciales desde una versión anterior de DB2 Spatial Extender.

#### Procedimiento

Para actualizar DB2 Spatial Extender a la Versión 9.7:

1. Actualice su servidor de DB2 de la versión 8, versión 9.1 o la versión 9.5 a la versión 9.7 realizando una de las tareas siguientes:

- “Actualización de un servidor de DB2(Windows)” en *Actualización a DB2 Versión 9.7*
- “Actualización de un servidor de DB2 (Linux y UNIX)” en *Actualización a DB2 Versión 9.7*

En la tarea de actualización, debe instalar DB2 Spatial Extender Versión 9.7 después de instalar DB2 Versión 9.7 para actualizar las instancias satisfactoriamente.

2. Finalice todas las conexiones con la base de datos.
3. Actualice las bases de datos espaciales desde la Versión 8, la Versión 9.1 o la Versión 9.5 a la Versión 9.7 utilizando el mandato db2se upgrade.

Compruebe en los archivos de mensajes si hay detalles sobre cualquier error que reciba. El archivo de mensajes también contiene información útil, como, por ejemplo, índices, vistas y la configuración de geocodificación que se ha actualizado.

---

## Actualización de DB2 Spatial Extender

Actualización de DB2 Spatial Extender Versión 9.7 de 32 bits a DB2 Spatial Extender Versión 9.7 de 64 bits requiere realizar una copia de seguridad de los índices espaciales, actualizar el servidor de DB2 a DB2 Versión 9.7 de 64 bits, instalar DB2 Spatial Extender Versión 9.7 de 64 bits y, a continuación, restaurar los índices espaciales de los que ha hecho copia de seguridad.

### Antes de empezar

- Asegúrese de que dispone de un espacio de tablas temporal del sistema con un tamaño de página de 8 KB o más y un tamaño mínimo de 500 páginas. Si no dispone de un espacio de tablas con estas características, consulte la sección sobre “creación de espacios de tablas temporales” en *Database Administration Concepts and Configuration Reference* para obtener más información sobre cómo crearlo.

### Acerca de esta tarea

Se realiza la actualización de DB2 Spatial Extender Versión 8, Versión 9.1 o 9.5 de 32 bits a DB2 Spatial Extender Versión 9.7 de 64 bits, deberá realizar la tarea “Actualización de DB2 Spatial Extender” en la página 27.

### Procedimiento

Para actualizar el servidor de DB2 Spatial Extender Versión 9.7 de 32 bits a DB2 Spatial Extender Versión 9.7 de 64 bits:

1. Haga una copia de seguridad de la base de datos.
2. Guarde los índices espaciales existentes emitiendo el mandato db2se save\_indexes desde un indicador de mandatos del sistema operativo.
3. Actualice su servidor de DB2 de Versión 8, Versión 9.1 o Versión 9.5 de 32 bits a DB2 Versión 9.7 de 64 bits realizando una de las tareas siguientes:
  - “Actualización de sus instancias de DB2 de 32 bits a instancias de 64 bits (Windows)” en *Instalación de servidores DB2*.
  - “Actualización de copias de DB2 (Linux y UNIX)” en *Database Administration Concepts and Configuration Reference*.
4. Instale DB2 Spatial Extender Versión 9.7.
5. Restablezca los índices espaciales emitiendo el mandato db2se restore\_indexes desde un indicador de mandatos del sistema operativo.

---

## Capítulo 6. Configuración de una base de datos

Este capítulo trata sobre cómo configurar una base de datos para albergar datos espaciales.

---

### Configuración de una base de datos para alojar datos espaciales

Este tema identifica los parámetros de configuración de DB2 que influyen en las operaciones de DB2 Spatial Extender.

DB2 Spatial Extender, que se ejecuta en el entorno de base de datos de DB2, funciona con la mayoría de valores de configuración por omisión de DB2. Sin embargo, algunos parámetros de configuración afectan a las operaciones espaciales. Debe ajustar estos parámetros de forma que las aplicaciones espaciales funcionen lo más eficazmente posible. Cuando modifique los valores de estos parámetros para una base de datos, el cambio sólo afectará a dicha base de datos. En algunos casos, es necesario seleccionar un valor distinto del valor por omisión para las operaciones espaciales. En otros casos, hacer esto es recomendable, en función de las aplicaciones y del entorno general de DB2.

Las secciones siguientes explican cómo ajustar el gestor de bases de datos de DB2 y los parámetros de configuración de base de datos que afectan a las operaciones de DB2 Spatial Extender.

### Ajuste de las características de anotaciones cronológicas de transacción

Antes de habilitar una base de datos para operaciones espaciales, asegúrese de tener suficiente capacidad para el registro de transacciones. Los valores por omisión para los parámetros de configuración de las anotaciones cronológicas de transacción no proporcionan suficiente capacidad de anotaciones cronológicas de transacción si piensa realizar alguna de las acciones siguientes:

- Habilitar una base de datos para operaciones espaciales en un entorno Windows
- Utilizar el procedimiento almacenado `ST_import_shape` para importar desde archivos de formas
- Utilizar la geocodificación con un ámbito de confirmación amplio
- Ejecutar transacciones simultáneas

Si piensa realizar alguna de estas acciones ahora o en el futuro, es necesario aumentar la capacidad de las anotaciones cronológicas de transacción para la base de datos aumentando uno o varios parámetros de configuración de las anotaciones cronológicas de transacción. Si éste no es su caso, puede utilizar las características por omisión.

**Recomendación:** Consulte la tabla siguiente para ver los valores mínimos recomendados para los tres parámetros de configuración de anotaciones cronológicas de transacción.

Tabla 1. Valores mínimos recomendados para los parámetros de configuración de transacción

Parámetro	Descripción	Valor por omisión	Valor mínimo recomendado
LOGFILSIZ	Especifica el tamaño del archivo de anotaciones cronológicas como un número de bloques de 4 KB	1000	1000
LOGPRIMARY	Especifica cuántos archivos de anotaciones cronológicas primarios se van a preasignar a los archivos de anotaciones cronológicas de recuperación	3	10
LOGSECOND	Especifica el número de archivos de anotaciones cronológicas secundarios	2	2

Si la capacidad de las anotaciones cronológicas de transacción no es la adecuada, se emitirá el siguiente mensaje de error cuando intente habilitar una base de datos para operaciones espaciales:

GSE0010N No hay suficiente espacio de anotaciones cronológicas disponible en DB2.

Para aumentar el valor de uno o varios parámetros de configuración:

1. Emita el mandato GET DATABASE CONFIGURATION para encontrar el valor actual de los parámetros LOGFILSIZ, LOGPRIMARY y LOGSECOND o bien ver la ventana **Configurar base de datos** del Centro de control de DB2.
2. Decida si desea cambiar uno, dos o tres de los valores como se indica en la tabla anterior.
3. Cambie cada valor que desee modificar. Puede cambiar los valores emitiendo uno o varios de los siguientes mandatos, donde *nombre\_bd* identifica la base de datos:

```
UPDATE DATABASE CONFIGURATION FOR nombre_bd USING LOGFILSIZ 1000
```

```
UPDATE DATABASE CONFIGURATION FOR nombre_bd USING LOGPRIMARY 10
```

```
UPDATE DATABASE CONFIGURATION FOR nombre_bd USING LOGSECOND 2
```

Si el único parámetro que cambia es LOGSECOND, el cambio tiene efecto inmediato.

Si cambia el parámetro LOGFILSIZ o LOGPRIMARY, o ambos:

1. Desconecte todas las aplicaciones de la base de datos.
2. Si la base de datos se ha activado explícitamente, desactive la base de datos.



Los cambios en los parámetros LOGFILSIZ o LOGPRIMARY, o en ambos, serán efectivos la próxima vez que se active la base de datos o que se establezca una conexión con la base de datos.

## Ajuste del tamaño de la pila de aplicaciones

Utilice el parámetro de configuración de la base de datos APPLHEAPSZ para especificar el tamaño de la pila de aplicaciones (en número de páginas de 4 KB). Este parámetro define el número de páginas de memoria privada que están disponibles para que las utilice el gestor de la base de datos en nombre de un agente o subagente específico. La pila se asigna cuando se inicializa un agente o subagente para una aplicación. La cantidad asignada es la cantidad mínima que es necesaria para procesar la petición dirigida al agente o subagente. Debido a que el agente o subagente necesita más espacio de pila para procesar sentencias de SQL más grandes, el gestor de bases de datos asigna memoria según sea necesario, hasta el máximo especificado en este parámetro. La pila de aplicaciones se asigna a partir de la memoria privada del agente.

El valor por omisión para el parámetro APPLHEAPSZ es 128 (páginas de 4 KB). Cuando ejecuta el procedimiento almacenado ST\_enable\_db, este valor debe ser como mínimo 2048.

**Recomendación:** En la mayoría de las aplicaciones DB2 Spatial Extender, especialmente aquellas que importan o exportan archivos de formas, utilice un valor de parámetro APPLHEAPSZ de al menos 2048.

Si APPLHEAPSZ está establecido en un valor inadecuado, se emitirá el siguiente mensaje de error cuando intente habilitar una base de datos para operaciones espaciales:

GSE0009N No hay espacio suficiente disponible en la pila de aplicaciones de DB2.

GSE0213N Una operación de vinculación no ha sido satisfactoria.  
SQLERROR = "SQL0001N La vinculación o la precompilación no se ha realizado satisfactoriamente. SQLSTATE=00000".

Para cambiar el tamaño de la pila de aplicaciones:

1. Emita el mandato GET DATABASE CONFIGURATION para encontrar el valor actual del parámetro APPLHEAPSZ o bien ver la ventana **Configurar base de datos** del DB2 Centro de control.
2. Cambie el valor por el valor recomendado de 2048 o bien por uno mayor. Puede cambiar el valor a 2048 emitiendo el siguiente mandato, donde *nombre\_bd* identifica la base de datos:  

```
UPDATE DATABASE CONFIGURATION FOR nombre_bd USING APPLHEAPSZ 2048
```
3. Desconecte todas las aplicaciones de la base de datos.
4. Si la base de datos se ha activado explícitamente, desactive la base de datos.

El cambio será efectivo la próxima vez que se active la base de datos o cuando se establezca una conexión con la base de datos.



---

## Capítulo 7. Configuración de recursos espaciales para una base de datos

Después de configurar la base de datos para alojar datos espaciales, está preparado para proporcionar a la base de datos recursos que necesitará al crear y gestionar columnas espaciales y al analizar datos espaciales. Estos recursos incluyen:

- Objetos proporcionados por Spatial Extender para dar soporte a operaciones espaciales; tales como, procedimientos almacenados para administrar una base de datos, tipos de datos espaciales y programas de utilidad para geocodificar e importar datos espaciales.
- Datos de referencia: Rangos de direcciones que DB2SE\_USA\_GEOCODER utiliza para convertir direcciones individuales en coordenadas.
- Cualquier codificador que los usuarios o los proveedores proporcionen.

Este capítulo describe estos recursos y presenta las tareas mediante las cuales el usuario los hace disponibles: habilitación de la base de datos para operaciones espaciales, configuración del acceso a datos de referencia y registro de geocodificadores que no son los geocodificadores por omisión.

---

### Cómo configurar recursos en la base de datos

La primera tarea que debe realizarse después de configurar la base de datos para alojar datos espaciales es hacer que la base de datos pueda dar soporte a operaciones espaciales; operaciones como, por ejemplo, llenar tablas con datos espaciales y procesar consultas espaciales. Esta tarea implica cargar la base de datos con ciertos recursos proporcionados por DB2 Spatial Extender. Esta sección describe estos recursos y describe brevemente la tarea.

### Inventario de recursos suministrados para la base de datos

Para habilitar una base de datos como soporte para las operaciones espaciales, DB2® Spatial Extender proporciona la base de datos con los siguientes recursos:

- Procedimientos almacenados. Cuando se solicita una operación espacial (por ejemplo, cuando se emite un mandato para importar datos espaciales) DB2 Spatial Extender invoca uno de estos procedimientos almacenados para realizar la operación.
- Tipos de datos espaciales. Debe asignar un tipo de datos espaciales a cada columna de tabla o de vista en la que se van a almacenar datos espaciales.
- Catálogo de DB2 Spatial Extender. Determinadas operaciones dependen de este catálogo. Por ejemplo, para poder acceder a la columna espacial desde las herramientas de visualización, la herramienta podría necesitar que la columna espacial esté registrada en el catálogo.
- Un índice reticular espacial. Le permite definir índices reticulares sobre columnas espaciales.
- Funciones espaciales. Sirven para trabajar con datos espaciales de diversas formas; por ejemplo, para determinar las relaciones entre geometrías y para generar más datos espaciales.
- Definiciones de sistemas de coordenadas.
- Sistemas de referencia espacial por omisión.

- Dos esquemas: DB2GSE y ST\_INFORMTN\_SCHEMA. DB2GSE contiene los objetos que se acaban de listar: procedimientos almacenados, tipos de datos espaciales, el catálogo de DB2 Spatial Extender, etc. Las vistas del catálogo están disponibles también en ST\_INFORMTN\_SCHEMA para cumplir el estándar SQL/MM.

## Habilitación de una base de datos para operaciones espaciales

### Antes de empezar

Antes de habilitar una base de datos para operaciones espaciales:

- Asegúrese de que su ID de usuario dispone de la autorización DBADM sobre la base de datos.
- Asegúrese de que dispone de un espacio de tablas temporal del sistema con un tamaño de página de 8 KB o más y un tamaño mínimo de 500 páginas. Si no dispone de un espacio de tablas con estas características, consulte la sección sobre “creación de espacios de tablas temporales” en *Database Administration Concepts and Configuration Reference* para obtener más información sobre cómo crearlo.

### Acerca de esta tarea

La tarea de que DB2 Spatial Extender proporcione una base de datos con recursos para crear columnas espaciales y manipular datos espaciales se suele denominar “habilitación de la base de datos para operaciones espaciales”.

### Procedimiento

Puede habilitar una base de datos para operaciones espaciales de cualquiera de estas maneras:

- Utilice la ventana Habilitar base de datos desde la opción de menú DB2 Spatial Extender. La opción de menú está disponible desde el objeto de base de datos del Centro de control de DB2.
- Emite el mandato `db2se enable_db`.
- Ejecute una aplicación que llame al procedimiento almacenado `db2gse.ST_enable_db`.

Puede seleccionar explícitamente el espacio de tabla donde desea que resida el catálogo de DB2 Spatial Extender. Si no lo hace, DB2 utilizará el espacio de tabla por omisión.

---

## Cómo trabajar con datos de referencia

Esta sección explica qué son los datos de referencia e indica lo que el usuario necesita hacer para poder acceder a los mismos.

### Datos de referencia

Los datos de referencia son rangos de direcciones que DB2SE\_USA\_GEOCODER utiliza para convertir direcciones individuales en coordenadas. Estos datos consisten en rangos de las direcciones más recientes que ha recopilado la Oficina del censo de Estados Unidos. Cuando DB2SE\_USA\_GEOCODER lee una dirección en la base de datos, busca los datos de referencia para:

- Los nombres de ciertas calles dentro de la zona designada por el código postal de la dirección. El geocodificador busca nombres que coincidan con el nombre de la calle contenida en la dirección en un grado especificado, o en un grado mayor que el especificado; por ejemplo, 80 por ciento o mayor.
- El rango de direcciones que corresponde con el número de la dirección.

Si se encuentra una coincidencia que no tiene el grado de concordancia solicitado, el geocodificador devuelve las coordenadas de la dirección que ha leído. Si no se encuentra una coincidencia o no tiene el grado de concordancia solicitado, el geocodificador devuelve un valor nulo.

Puede utilizar un archivo de configuración avanzada, llamado *archivo localizador*, para actuar más sobre el proceso realizado por el geocodificador, DB2SE\_USA\_GEOCODER. Normalmente, no es necesario modificar la configuración predeterminada proporcionada por DB2® Spatial Extender en este archivo.

## Configuración del acceso a los datos de referencia para DB2SE\_USA\_GEOCODER

Los datos de consulta para DB2SE\_USA\_GEOCODER están disponibles para su descarga. Esta sección describe cómo preparar el acceso a los datos de consulta de geocodificador.

### Antes de empezar

Asegúrese de disponer de suficiente espacio para contener los datos de consulta del geocodificador (aproximadamente 700 MB).

### Procedimiento

Para configurar el acceso a los datos de referencia para DB2SE\_USA\_GEOCODER:

1. Descargue el archivo zip de datos de consulta del geocodificador seleccionando la oferta DB2 Spatial Extender Sample Map Data en la página web de pruebas y demostraciones, disponible en <http://www.ibm.com/software/data/spatial/db2spatial>, y extraiga los archivos del zip en uno de los directorios siguientes:
  - `$DB2INSTANCE/sql1lib/gse/refdata/` en sistemas operativos Linux o UNIX.
  - `%DB2PATH%\gse\refdata\` en sistemas operativos Windows.

Si desea más información, consulte el archivo README que acompaña a los datos de referencia.

2. Indique a DB2SE\_USA\_GEOCODER el nombre y la ubicación del archivo localizador y el mapa base. Haga esto configurando con los valores correctos los parámetros `base_map` y `locator_file` de DB2SE\_USA\_GEOCODER. Para más información, consulte al administrador de la base de datos o póngase en contacto con el representante de IBM.

## Registro de un geocodificador

DB2SE\_USA\_GEOCODER se registra automáticamente en DB2 Spatial Extender cuando se habilita una base de datos para operaciones espaciales. Antes de que se puedan utilizar otros geocodificadores, éstos también se deben registrar.

### Antes de empezar

Antes de poder registrar un geocodificador, el ID de usuario debe tener autoridad DBADM sobre la base de datos en la que se ubica el geocodificador.

### **Procedimiento**

Puede registrar un geocodificador de cualquiera de las maneras siguientes:

- Regístrelo desde la ventana Registrar geocodificador del Centro de control de DB2.
- Emita el mandato `db2se register_gc`.
- Ejecute una aplicación que llame al procedimiento almacenado `db2gse.ST_register_geocoder`.

---

## Capítulo 8. Configuración de recursos espaciales para un proyecto

Después de habilitar la base de datos para operaciones espaciales, está preparado para crear proyectos que utilicen datos espaciales. Entre los recursos que cada proyecto necesita está un sistema de coordenadas al que se adecuan los datos espaciales y un sistema de referencia espacial que define la extensión del área geográfica a la que los datos hacen referencia. Este capítulo:

- Trata la naturaleza de los sistemas de coordenadas e indica cómo crearlos
- Explica qué son los sistemas de referencia espacial e indica como crearlos

---

### Cómo utilizar sistemas de coordenadas

Cuando planea un proyecto que utiliza datos espaciales, es necesario determinar si los datos se deben basar en uno de los sistemas de coordenadas que están registrados en el catálogo de Spatial Extender. Si ninguno de estos sistemas de coordenadas cumple los requisitos, puede crear uno que lo haga. Esta explicación explica el concepto de sistemas de coordenadas y presenta las tareas de seleccionar uno para utilizarlo y de crear uno nuevo.

### Sistemas de coordenadas

Un sistema de coordenadas es un marco para definir las ubicaciones relativas de los objetos en un área determinada; por ejemplo, un área en la superficie de la tierra o la superficie de la tierra en su totalidad. DB2<sup>®</sup> Spatial Extender da soporte a los siguientes tipos de sistemas de coordenadas para determinar la ubicación de un elemento geográfico:

#### Sistema de coordenadas geográficas

Un sistema de *coordenadas geográficas* es aquél que utiliza una superficie esférica tridimensional para determinar ubicaciones en la tierra. Se puede hacer referencia a cualquier ubicación de la tierra mediante un punto con coordenadas de latitud y longitud basadas en unidades angulares de medida.

#### Sistema de coordenadas proyectadas

Un *sistema de coordenadas proyectadas* es una representación bidimensional plana de la tierra. Utilice coordenadas rectilíneas (cartesianas) basadas en unidades lineales de medida. Se basa en un modelo esférico (o esferoidal) de la tierra y sus coordenadas se relacionan con coordenadas geográficas mediante una transformación de proyección.

### Sistema de coordenadas geográficas

Un *sistema de coordenadas geográficas* es aquel que utiliza una superficie esférica tridimensional para determinar ubicaciones en la tierra. Se puede hacer referencia a cualquier ubicación de la tierra mediante un punto con coordenadas de longitud y latitud. Los valores de los puntos pueden tener las siguientes unidades de medida:

- Las unidades lineales cuando el sistema de coordenadas geográficas tiene un identificador de sistema de referencia espacial (SRID) que DB2<sup>®</sup> Geodetic Data Management Feature reconoce.

- Cualquiera de las unidades siguientes cuando el sistema de coordenadas geográficas tiene un SRID que DB2 Geodetic Data Management Feature no reconoce.
  - Grados decimales
  - Minutos decimales
  - Segundos decimales
  - Gradianes
  - Radianes

Por ejemplo, en la Figura 6 se muestra un sistema de coordenadas geográficas en el que una ubicación se representa mediante las coordenadas de 80 grados este de longitud y de 55 grados norte de latitud.

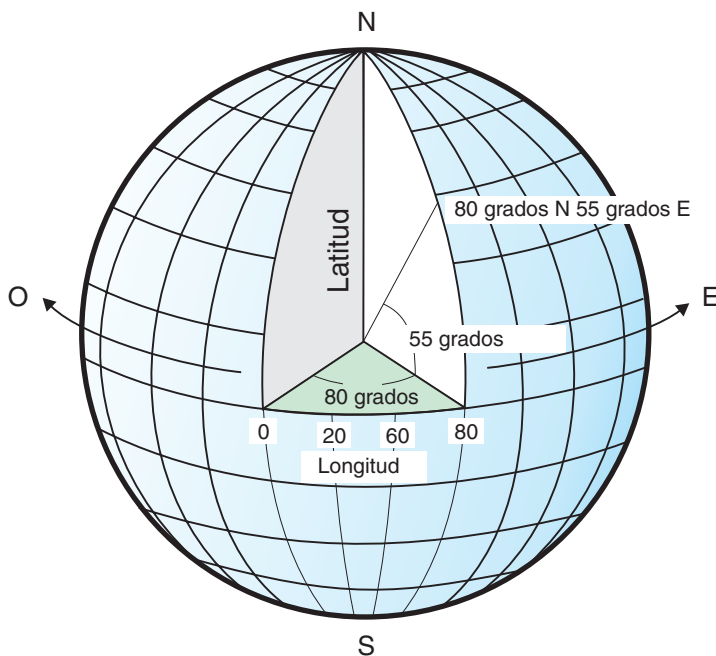


Figura 6. Un sistema de coordenadas geográficas

Las líneas que van del este al oeste tienen cada una un valor de latitud constante y se denominan *paralelos*. Son equidistantes y paralelas entre ellas, y forman círculos concéntricos alrededor de la tierra. El *ecuador* es el círculo más grande y divide la tierra por la mitad. Es equidistante de cada uno de los polos y el valor de esta latitud es cero. Las ubicaciones situadas al norte del ecuador tienen latitudes positivas entre 0 y +90 grados, mientras que las ubicaciones al sur del ecuador tienen latitudes negativas comprendidas entre 0 y -90 grados.

La Figura 7 en la página 39 ilustra las líneas de latitud.



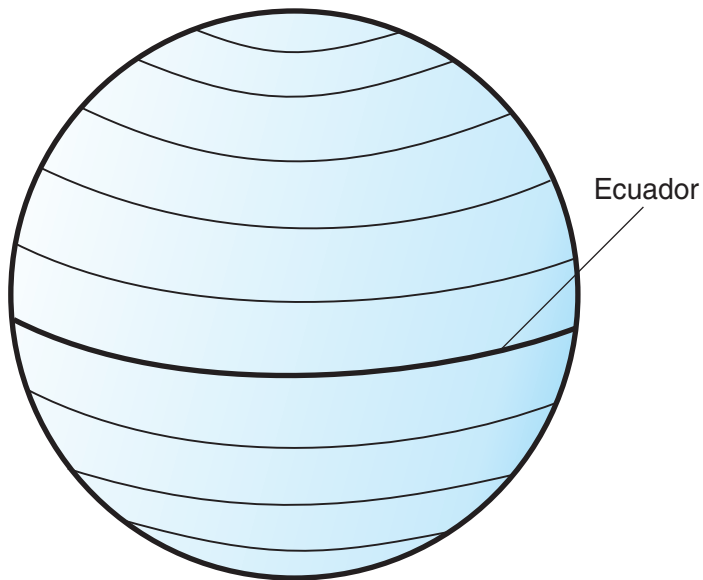


Figura 7. Líneas de latitud

Las líneas que van al norte y al sur tienen un valor de longitud constante y se denominan *meridianos*. Forman círculos del mismo tamaño alrededor de la tierra y forman intersecciones en los polos. El *meridiano de origen* es la línea de longitud que define el origen (cero grados) de las coordenadas de longitud. Una de las ubicaciones del meridiano de origen utilizadas más habitualmente es la línea que pasa por Greenwich, Inglaterra. Sin embargo, también se han utilizado otras líneas de longitud, como las que pasan por Berna, Bogotá y París, como meridiano de origen. Las ubicaciones situadas al este del meridiano de origen hasta su meridiano *antípoda* (la continuación del meridiano de origen en el otro lado del globo) tienen longitudes positivas comprendidas entre 0 y +180 grados. Las ubicaciones situadas al oeste del meridiano de origen tienen longitudes negativas comprendidas entre 0 y -180 grados.

La Figura 8 en la página 40 muestra las líneas de longitud.

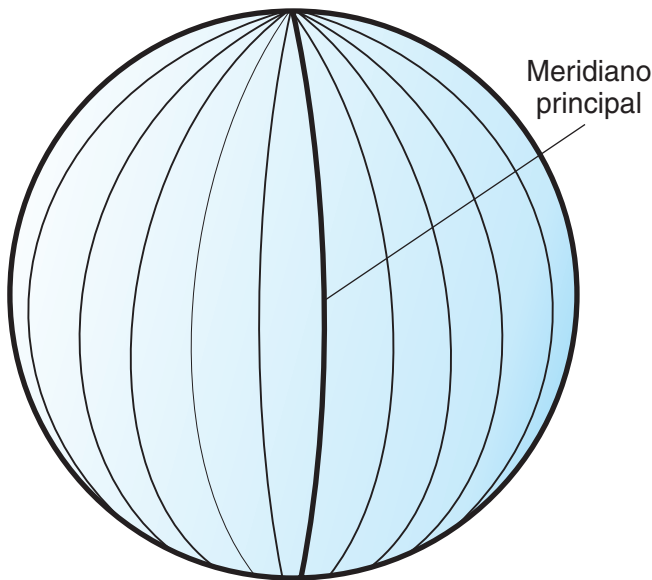


Figura 8. Líneas de longitud

Las líneas de latitud y longitud pueden recubrir el globo para formar una cuadrícula denominada, a *red geográfica*. El punto de origen de la red geográfica es (0,0), donde el ecuador y el meridiano de origen forman intersección. El ecuador es el único lugar en la red geográfica donde la distancia lineal correspondiente a un grado de latitud equivale aproximadamente a la distancia correspondiente a un grado de longitud. Debido a que las líneas de longitud convergen en los polos, la distancia entre dos meridianos es diferente en cada paralelo. Por lo tanto, a medida que nos acercamos a los polos, la distancia correspondiente a un grado de latitud será superior a la que corresponde a un grado de longitud.

También es difícil determinar las longitudes de las líneas de latitud utilizando la red geográfica. Las líneas de latitud son círculos concéntricos que se hacen más pequeños cerca de los polos. Forman un único punto en los polos donde empiezan los meridianos. En el ecuador, un grado de longitud equivale aproximadamente a 111.321 kilómetros, mientras que a 60 grados de latitud, un grado de longitud equivale a sólo 55.802 km (esta aproximación se basa en el esferoide Clarke 1866). Por consiguiente, como no existe una longitud uniforme de grados de latitud y longitud, la distancia entre los puntos no se puede medir con precisión utilizando unidades angulares de medida.

La Figura 9 en la página 41 muestra las diferentes dimensiones entre ubicaciones en la red geográfica.

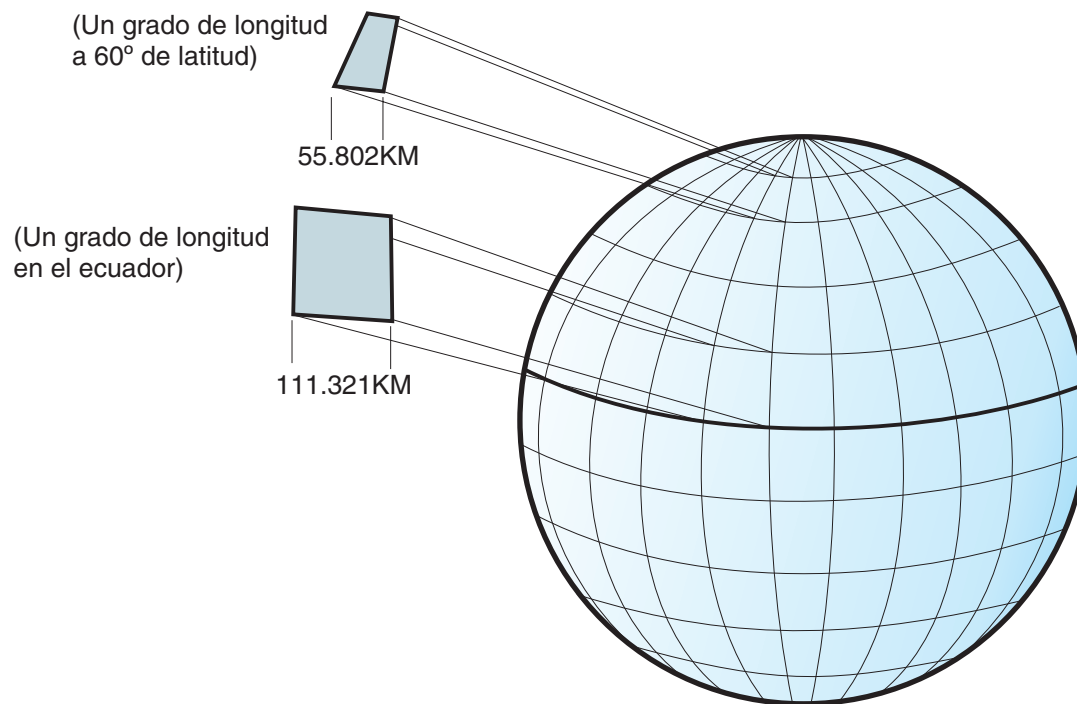
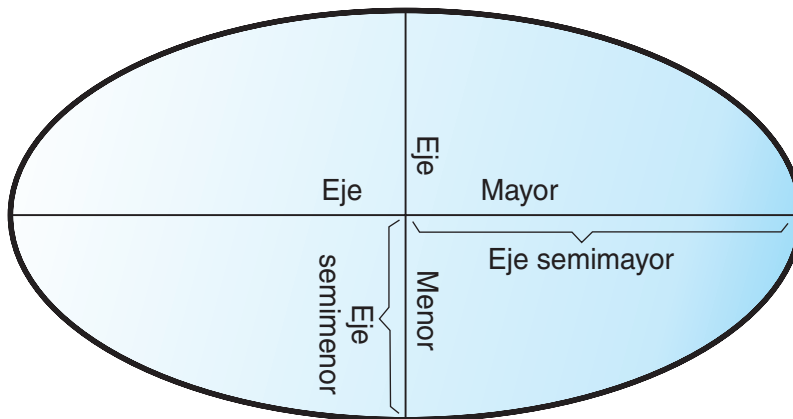
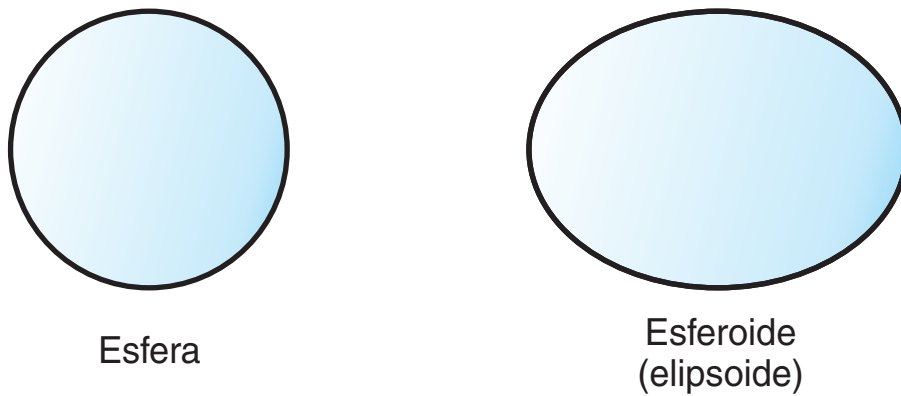


Figura 9. Diferentes dimensiones entre ubicaciones en la red geográfica

Un sistema de coordenadas se puede definir mediante una aproximación de esfera o esferoide de la forma de la tierra. Debido a que la tierra no es completamente redonda, un esferoide puede mantener la precisión de un mapa, dependiendo de la ubicación de la tierra. Un *esferoide* es un elipsoide, que está basado en una elipse, mientras que una esfera está basada en un círculo.

La forma de la elipse está determinada por dos radios. El radio mayor se denomina eje semimayor y el radio menor se denomina eje semimenor. Un elipsoide es una forma tridimensional generada al girar una elipse alrededor de uno de sus ejes.

En la Figura 10 en la página 42 se muestran las aproximaciones de esfera y esferoide de la tierra y los ejes mayor y menor de una elipse.



### Los ejes mayor y menor de una elipse

Figura 10. Aproximaciones de esfera y esferoide

Un *sistema de referencia* es un conjunto de valores que define la posición del esferoide con relación al centro de la tierra. El sistema de referencia proporciona un marco de referencia para medir ubicaciones y define el origen y la orientación de las líneas de latitud y longitud. Algunos sistemas de referencia son globales y pretenden proporcionar una buena precisión media en todo el mundo. Un sistema de referencia local alinea su esferoide para que se ajuste con precisión a la superficie de la tierra en una zona determinada. Por lo tanto, las mediciones del sistema de coordenadas no serán precisas si se utilizan con un área distinta del área para la que se han diseñado.

En la Figura 11 en la página 43 se muestra cómo sistemas de referencia diferentes se alinean con la superficie de la tierra. El sistema de referencia local, NAD27, se alinea más estrechamente con la superficie de la tierra que el sistema de referencia centrado en la tierra, WGS84, en esta ubicación específica.

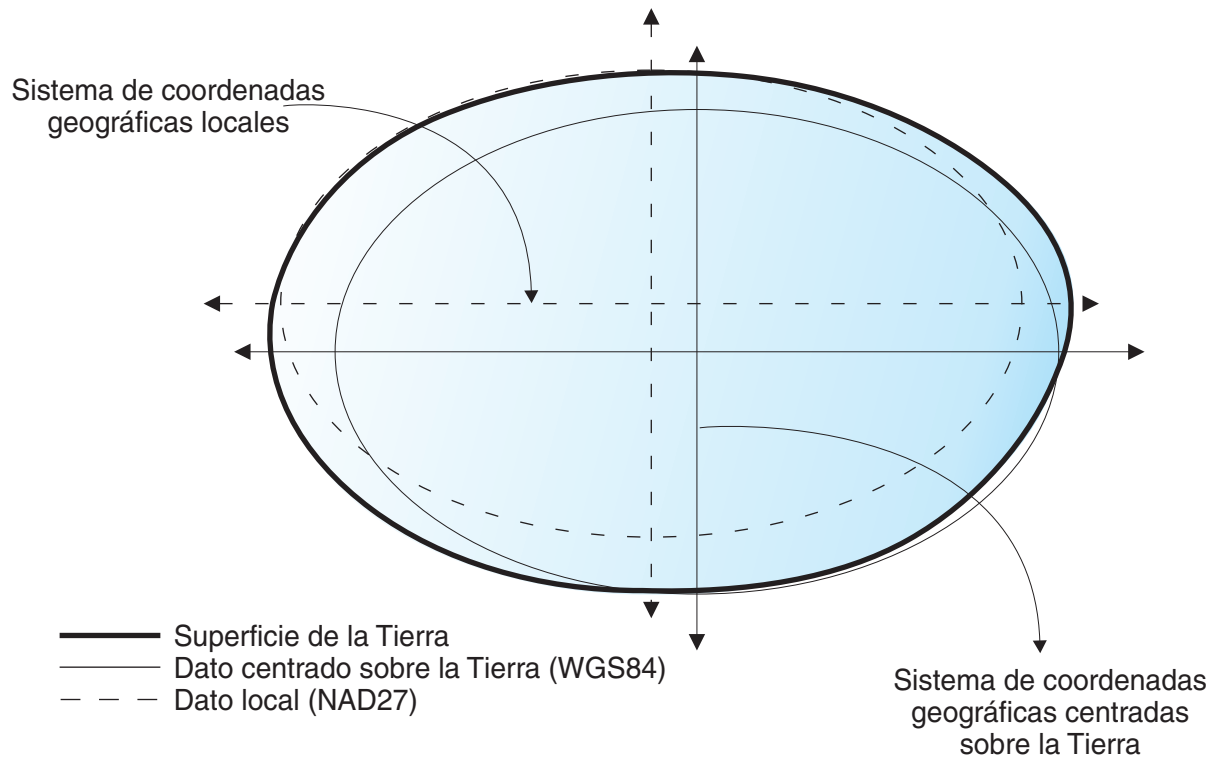


Figura 11. Alineaciones de sistemas de referencia

Siempre que cambie el sistema de referencia, el sistema de coordenadas geográficas se modificará y los valores de coordenadas cambiarán. Por ejemplo, las coordenadas en DMS de un punto de control situado en Redlands, California, utilizando el sistema de referencia NAD 1983 (North American Datum de 1983) son: "-117 12 57.75961 34 01 43.77884". Las coordenadas del mismo punto en el sistema de referencia NAD 1927 (North American Datum de 1927) son: "-117 12 54.61539 34 01 43.72995".

## Sistemas de coordenadas proyectadas

Un *sistema de coordenadas proyectadas* es una representación plana, bidimensional de la tierra. Se basa en un sistema de coordenadas geográficas esféricas o esferoidales, pero utiliza unidades lineales para las coordenadas, de forma que los cálculos de distancia y área se pueden realizar fácilmente en términos de esas mismas unidades.

Las coordenadas de longitud y latitud se convierten en coordenadas  $x$ ,  $y$  en la proyección plana. La coordenada  $x$  representa normalmente la orientación hacia el este de un punto y la coordenada  $y$  representa normalmente la dirección hacia el norte de un punto. La línea central que va de este a oeste se denomina eje  $x$ , y la línea central que va del norte al sur se denomina eje  $y$ .

La intersección de los ejes  $x$  e  $y$  es el origen y normalmente tiene la coordenada  $(0,0)$ . Los valores situados por encima del eje  $x$  son positivos y los valores situados por debajo del eje  $x$  son negativos. Las líneas paralelas al eje  $x$  son equidistantes entre ellas. Los valores situados a la derecha del eje  $y$  son positivos y los valores situados a la izquierda del eje  $y$  son negativos. Las líneas paralelas al eje  $y$  son equidistantes.

Se utilizan fórmulas matemáticas para convertir un sistema de coordenadas geográficas tridimensionales en un sistema de coordenadas proyectadas planas bidimensionales. La transformación se denomina *proyección cartográfica*. Las proyecciones cartográficas normalmente se clasifican según la superficie de proyección utilizada, como, por ejemplo, superficies cónicas, cilíndricas y planas. En función de la proyección utilizada, diferentes propiedades espaciales aparecerán distorsionadas. Las proyecciones están diseñadas para minimizar la distorsión de una o dos características de datos, pero es posible que la distancia, el área, la forma, la dirección o una combinación de estas propiedades no sean representaciones precisas de los datos de los que se está realizando el modelo. Existen varios tipos de proyecciones disponibles. Mientras que la mayoría de las proyecciones cartográficas intentan conservar cierta precisión de las propiedades espaciales, otras en su lugar intentan minimizar la distorsión general, como por ejemplo la proyección *Robinson*. Los tipos más habituales de proyecciones cartográficas incluyen los siguientes:

#### **Proyecciones de áreas iguales**

Estas proyecciones conservan el área de elementos específicos. Estas proyecciones distorsionan la forma, el ángulo y la escala. La proyección *cónica de áreas iguales Albers* es un ejemplo de una proyección de áreas iguales.

#### **Proyecciones conformes**

Estas proyecciones conservan la forma local de áreas pequeñas. Estas proyecciones conservan ángulos individuales para describir relaciones espaciales mostrando líneas perpendiculares de red geográfica que forman intersección en ángulos de 90 grados en el mapa. Se conservan todos los ángulos; sin embargo, el área del mapa está distorsionada. Las proyecciones *Mercator* y *Cónica conforme de Lambert* son ejemplos de proyecciones conformes.

#### **Proyecciones equidistantes**

Estas proyecciones conservan las distancias entre ciertos puntos manteniendo la escala de un conjunto de datos determinado. Algunas de las distancias serán distancias verdaderas, que son las mismas distancias en la misma escala que el globo. Si sale fuera del conjunto de datos, la escala se distorsionará más. La proyección *sinusoidal* y la proyección *cónica equidistante* son ejemplos de proyecciones equidistantes.

#### **Proyecciones de dirección verdadera o azimutales**

Estas proyecciones conservan la dirección de un punto a otros puntos manteniendo algunos de los arcos de círculo grandes. Estas proyecciones dan correctamente las direcciones o azimuts de todos los puntos del mapa respecto al centro. Los mapas azimutales se pueden combinar con proyecciones de áreas iguales, conformes y equidistantes. La proyección *azimutal de igual área de Lambert* y la proyección *azimutal equidistante* son ejemplos de proyecciones azimutales.

## **Selección o creación de sistemas de coordenadas**

Un primer paso en la planificación de un proyecto es determinar qué sistema de coordenadas se debe utilizar.

Antes de crear un sistema de coordenadas, el ID de usuario debe tener autorización DBADM sobre la base de datos que se ha habilitado para operaciones espaciales. No se necesita ninguna autorización para utilizar un sistema de coordenadas existente.

Después de habilitar una base de datos para operaciones espaciales, está listo para planear proyectos que utilicen datos espaciales. Puede utilizar un sistema de coordenadas proporcionado con DB2 Spatial Extender o uno que haya sido creado por otro usuario. Con DB2 Spatial Extender se proporcionan más de 2000 sistemas de coordenadas. Entre éstos se encuentran:

- Un sistema de coordenadas al que DB2 Spatial Extender se refiere como Sin especificar. Utilice este sistema de coordenadas cuando:
  - Necesite definir ubicaciones que no tengan una relación directa con la superficie de la tierra; por ejemplo, ubicaciones de oficinas situadas dentro de un edificio de oficinas o ubicaciones de estanterías situadas dentro de un almacén.
  - Puede definir estas ubicaciones en términos de coordenadas positivas que incluyan pocos valores decimales o ninguno.
- GCS\_NORTH\_AMERICAN\_1983. Utilice este sistema de coordenadas cuando necesite definir ubicaciones en los Estados Unidos; por ejemplo:
  - Cuando importe datos espaciales para Estados Unidos desde los datos de mapa espacial disponibles para descargar.
  - Cuando desee utilizar el geocodificador proporcionado con DB2 Spatial Extender para geocodificar direcciones en Estados Unidos.

Para saber más sobre los sistemas de coordenadas y para determinar qué otros sistemas de coordenadas se proporcionan con DB2 Spatial Extender y qué otros sistemas de coordenadas (si los hay) han sido creados por otros usuarios, consulte la vista de catálogo DB2SE.ST\_COORDINATE\_SYSTEMS.

Para realizar esta tarea:

Elija qué método desea utilizar para crear un sistema de coordenadas:

- Créelo desde la ventana Crear sistema de coordenadas del Centro de control de DB2.
- Emita el mandato `db2se create_cs` desde el procesador de línea de mandatos `db2se`.
- Ejecute una aplicación que invoque al procedimiento almacenado `db2se.ST_create_coordsys`.

---

## Cómo configurar sistemas de referencia espacial

Cuando planea un proyecto que utiliza datos espaciales, es necesario determinar si alguno de los sistemas de referencia espacial disponible para el usuario se puede utilizar para estos datos. Si ninguno de los sistemas disponibles es adecuado para los datos, puede crear uno que lo sea. Esta sección explica el concepto de sistemas de referencia espacial y describe las tareas de seleccionar cuál utilizar y de crear uno.

### Sistemas de referencia espacial

Un *sistema de referencia espacial* es un conjunto de parámetros que incluye lo siguiente:

- El nombre del sistema de coordenadas del cual se obtienen las coordenadas.
- El identificador numérico que identifica de forma exclusiva al sistema de referencia espacial.
- Coordenadas que definen la máxima extensión de espacio posible a la que se hace referencia mediante un rango determinado de coordenadas.

- Números que, cuando se utilizan en ciertas operaciones matemáticas, convierten las coordenadas recibidas como datos de entrada en valores que se pueden procesar con máxima eficacia.

Las siguientes secciones abordan los valores de parámetros que definen un identificador, una extensión de espacio máximo y factores de conversión.

## **Identificador de sistemas de referencia espacial**

El identificador de sistemas de referencia espacial (SRID) se utiliza como parámetro de entrada para varias funciones espaciales.

Para un sistema de referencia espacial, el valor del SRID debe estar comprendido entre 2000000000 y 2000001000. DB2<sup>®</sup> Geodetic Data Management Feature proporciona 318 sistemas de referencia espacial geodésicos (SRS).

## **Definición del espacio que comprende coordenadas almacenadas en una columna espacial**

Las coordenadas de una columna espacial definen normalmente ubicaciones que se extienden a lo largo de la tierra. El espacio que cubre la extensión (de este a oeste y de norte a sur) se denomina *extensión espacial*. Por ejemplo, imagine una planicie aluvial cuyas coordenadas se almacenan en una columna espacial. Supongamos que las coordenadas situadas más al oeste y más al este son valores de latitud de -24,556 y -19,338, respectivamente, y que las coordenadas situadas más al norte y más al sur son valores de longitud de 18,819 y 15,809 grados, respectivamente. La extensión espacial de la planicie aluvial es un espacio que se extiende en un plano oeste-este entre las dos latitudes y en un plano norte-sur entre las dos longitudes. Puede incluir estos valores en un sistema de referencia espacial asignándolos a ciertos parámetros. Si la columna espacial incluye medidas y coordenadas Z, necesitará incluir también las coordenadas y las medidas Z más altas y más bajas en el sistema de referencia espacial.

El término extensión espacial se puede referir no solamente a una extensión real de ubicaciones, como en el párrafo anterior, sino también a una extensión potencial. Imagine que las planicies aluviales del ejemplo anterior se ensancharan en los próximos cinco años. Podría calcular cuáles serían al final del quinto año las coordenadas situadas más al oeste, este, norte y sur de las planicies. A continuación, podría asignar estos cálculos, en lugar de las coordenadas actuales, a los parámetros para una extensión espacial. De esta manera, podría conservar el sistema de referencia espacial conforme las planicies se ensanchen y sus latitudes y longitudes mayores se añadan a la columna espacial. En caso contrario, si el sistema de referencia espacial estuviera limitado a las latitudes y longitudes originales, sería necesario modificarlo o sustituirlo conforme las planicies aluviales crecieran.

## **Conversión a valores que mejoran el rendimiento**

Normalmente, la mayoría de coordenadas en un sistema de coordenadas son valores decimales; algunos son números enteros. Además, las coordenadas situadas al este del origen son valores positivos; aquellas situadas al oeste son valores negativos. Antes de ser almacenadas por Spatial Extender, las coordenadas con valores negativos se convierten a valores positivos y las coordenadas con valores decimales se convierten en números enteros. Como resultado, Spatial Extender almacena todas las coordenadas como valores enteros positivos. El propósito de esta conversión es mejorar el rendimiento al procesar las coordenadas.



Algunos parámetros de un sistema de referencia espacial se utilizan para realizar las conversiones que se describen en el párrafo anterior. Un parámetro, denominado *desplazamiento*, se resta de cada coordenada negativa, lo que deja un valor positivo como resultado. Cada coordenada decimal se multiplica por otro parámetro denominado *factor de escala*, lo que da como resultado un entero cuya precisión es la misma que la de la coordenada decimal. (El desplazamiento se sustrae de las coordenadas positivas así como de las negativas; y las coordenadas no decimales, así como las coordenadas decimales, se multiplican por el factor de escala. De esta forma, todas las coordenadas positivas y no decimales permanecen en proporción con las coordenadas negativas y decimales.)

Estas conversiones se realizan internamente y sólo permanecen vigentes hasta que se recuperen las coordenadas. Los datos de entrada y los resultados de las consultas siempre contienen coordenadas en su forma original y no convertida.

## **Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo**

Después de determinar el sistema de coordenadas que utilizará, ya está preparado para proporcionar un sistema de referencia espacial adecuado para los datos de coordenadas con los que está trabajando. DB2 Spatial Extender proporciona cinco sistemas de referencia espacial para datos espaciales y DB2 Geodetic Data Management Feature proporciona 318 sistemas de referencia espacial geodésicos para datos geodésicos.

Responda a las siguientes preguntas para determinar si puede utilizar uno de los sistemas de referencia espacial por omisión o los sistemas de referencia geodésicos predefinidos.

1. ¿El sistema de coordenadas en el que está basado el sistema de referencia espacial por omisión cubre la zona geográfica con la que está trabajando? Estos sistemas de coordenadas se muestran en el apartado “Sistemas de referencia espacial suministrados con DB2 Spatial Extender” en la página 48.
2. ¿Se encuentran los datos en un sistema de coordenadas geográficas que utiliza Grados decimales o grads como unidades de medida? ¿Los datos abarcan una porción grande de la superficie de la tierra? ¿Necesita realizar cálculos precisos sobre distancia, longitud y área? ¿Alguno de los datos se encuentra cerca del polo norte, del polo sur o de la línea de datos internacional? Si ha contestado afirmativamente a alguna de estas preguntas, probablemente deseará utilizar uno de los 318 sistemas de referencia espacial geodésicos. Para obtener información sobre estos sistemas de referencia espacial geodésicos predefinidos, consulte el apartado “Sistemas de referencia soportados por DB2 Geodetic Data Management Feature” en la página 184.
3. ¿Los factores de conversión asociados a uno de los sistemas de referencia espacial por omisión funcionan con datos de coordenadas?

Spatial Extender utiliza valores de desplazamiento y escala para convertir los datos de coordenadas proporcionados en enteros positivos. Para determinar si sus datos de coordenadas funcionan con los valores de desplazamiento y los factores de escala proporcionados para uno de los sistemas de referencia espacial por omisión:

- a. Revise la información en el apartado “Factores de conversión que transforman datos de coordenadas en enteros” en la página 50.

- b. Observe cómo están definidos estos factores para los sistemas de referencia espacial por omisión. Si, después de aplicar el valor de desplazamiento a las coordenadas X e Y mínimas, ninguna de estas dos coordenadas es superior a 0, deberá crear un sistema de referencia espacial nuevo y definir los desplazamientos por sí mismo. Para obtener más información sobre cómo crear un sistema de referencia espacial nuevo, consulte el apartado “Creación de un sistema de referencia espacial” en la página 52.
4. ¿Los datos con los que está trabajando incluyen coordenadas de altura y profundidad (coordenadas Z) o medidas (coordenadas M)? Si trabaja con coordenadas Z o M, es posible que deba crear un nuevo sistema de referencia espacial factores de escala y desplazamientos Z o M adecuados para sus datos.
5. Si los sistemas de referencia espacial o los sistemas de referencia geodésicos no funcionan con los datos, deberá acceder al apartado “Creación de un sistema de referencia espacial” en la página 52.

Después de decidir qué sistema de referencia espacial necesita, deberá especificar su elección a Spatial Extender cuando realice una de las siguientes tareas:

- “Sistemas de referencia espacial suministrados con DB2 Spatial Extender”
- “Sistemas de referencia soportados por DB2 Geodetic Data Management Feature” en la página 184

## **Sistemas de referencia espacial suministrados con DB2 Spatial Extender**

El sistema de referencia espacial convierte los datos de coordenadas en enteros positivos.

DB2 Spatial Extender proporciona los sistemas de referencia espacial que se muestran en la tabla siguiente, junto con el sistema de coordenadas en el que se basa cada sistema de referencia espacial y los valores de desplazamiento y los factores de escala que DB2 Spatial Extender utiliza para convertir los datos de coordenadas en enteros positivos. Encontrará información sobre estos sistemas de referencia espacial en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS.

Si trabaja con grados decimales, los valores de desplazamiento y los factores de escala de los sistemas de referencia espacial por omisión darán soporte a una amplia gama de coordenadas de latitud-longitud y conservarán 6 posiciones decimales, equivalente a 10 cm aproximadamente. Los datos del mapa espacial de ejemplo y los datos de consulta del geocodificador están en grados decimales.

Si piensa utilizar el geocodificador, que sólo funciona con direcciones de los EE.UU., asegúrese de seleccionar o crear un sistema de referencia espacial que maneje coordenadas de los EE.UU., como por ejemplo el sistema de coordenadas GCS\_NORTH\_AMERICAN\_1983. Si no especifica de qué sistema de coordenadas deberían derivarse los datos espaciales, Spatial Extender utiliza el sistema de referencia espacial DEFAULT\_SRS.

Si ninguno de los sistemas de referencia espacial por omisión se ajusta a sus necesidades, puede crear un sistema de referencia espacial nuevo.

Tabla 2. Sistemas de referencia espacial proporcionados con DB2 Spatial Extender

Sistema de referencia espacial	ID de SRS	Sistema de coordenadas	Valores de desplazamiento	Factores de escala	Cuándo se utiliza
DEFAULT_SRS	0	Ninguno	xOffset = 0 yOffset = 0 zOffset = 0 mOffset = 0	xScale = 1 yScale = 1 zScale = 1 mScale = 1	Puede seleccionar este sistema cuando sus datos sean independientes de un sistema de coordenadas o no pueda o no necesite especificar uno.
NAD83_SRS_1	1	GCS_NORTH_AMERICAN_1983	xOffset = -180 yOffset = -90 zOffset = 0 mOffset = 0	xScale = 1,000,000 yScale = 1,000,000 zScale = 1 mScale = 1	Puede seleccionar este sistema de referencia espacial si va a utilizar los datos de ejemplo de EE.UU. suministrados con DB2 Spatial Extender. Si los datos de coordenadas con los que está trabajando se recopilaron después de 1983, utilice este sistema en lugar de NAD27_SRS_1002.
NAD27_SRS_1002	1002	GCS_NORTH_AMERICAN_1927	xOffset = -180 yOffset = -90 zOffset = 0 mOffset = 0	xScale = 5,965,232 yScale = 5,965,232 zScale = 1 mScale = 1	Puede seleccionar este sistema de referencia espacial si va a utilizar los datos de ejemplo de EE.UU. suministrados con DB2 Spatial Extender. Si los datos de coordenadas con los que está trabajando se recopilaron antes de 1983, utilice este sistema en lugar de NAD83_SRS_1. Este sistema proporciona un mayor grado de precisión que el resto de sistemas de referencia espacial por omisión.

Tabla 2. Sistemas de referencia espacial proporcionados con DB2 Spatial Extender (continuación)

Sistema de referencia espacial	ID de SRS	Sistema de coordenadas	Valores de desplazamiento	Factores de escala	Cuándo se utiliza
WGS84_SRS_1003	1003	GCS_WGS_1984	xOffset = -180 yOffset = -90 zOffset = 0 mOffset = 0	xScale = 5,965,232 yScale = 5,965,232 zScale = 1 mScale = 1	Puede seleccionar este sistema de referencia espacial si trabaja con datos que no sean de los EE.UU. (Este sistema maneja coordenadas de todo el mundo). No utilice este sistema si va a utilizar el geocodificador por omisión suministrado con DB2 Spatial Extender, porque el geocodificador sólo sirve para direcciones de EE.UU.
DE_HDN_SRS_1004	1004	GCSW_DEUTSCHE_HAUPTDREIECKSNETZ	xOffset = -180 yOffset = -90 zOffset = 0 mOffset = 0	xScale = 5,965,232 yScale = 5,965,232 zScale = 1 mScale = 1	Este sistema de referencia espacial está basado en un sistema de coordenadas para direcciones alemanas.

## Factores de conversión que transforman datos de coordenadas en enteros

DB2 Spatial Extender utiliza valores de desplazamiento y factores de escala para convertir los datos de coordenadas proporcionados en enteros positivos. Los sistemas de referencia espacial por omisión ya tienen valores de desplazamiento y factores de escala asociados a los mismos. Si crea un nuevo sistema de referencia espacial, determine los factores de escala y, opcionalmente, los valores de desplazamiento que funcionen mejor con sus datos.

### Valores de desplazamiento

Un valor de desplazamiento es un número que se resta de todas las coordenadas, dejando sólo valores positivos como resto. Spatial Extender convierte los datos de coordenadas utilizando las siguientes fórmulas para asegurarse de que todos los valores de coordenadas ajustadas son mayores que 0.

**Notación de la fórmula:** En estas fórmulas, la notación “mín” representa “el mínimo de todos”. Por ejemplo, “mín(x)” significa “el mínimo de todas las

coordenadas  $x''$ . El desplazamiento de cada dirección geográfica se representa como `dimensionOffset`. Por ejemplo, `xOffset` es el valor de desplazamiento aplicado a todas las coordenadas X.

```
min(x) - xOffset = 0  
min(y) - yOffset = 0  
min(z) - zoffset = 0  
min(m) - moffset = 0
```

## Factores de escala

Un factor de escala es un valor que, multiplicado por medidas y coordenadas decimales, da lugar a números enteros que tienen como mínimo el mismo número de dígitos significantes que las medidas y coordenadas originales. `Spatial Extender` convierte las coordenadas decimales utilizando las siguientes fórmulas para garantizar que todos los valores de coordenadas ajustados sean números enteros positivos. Los valores convertidos no pueden sobrepasar  $2^{53}$  (aproximadamente  $9 * 10^{15}$ ).

**Notación de la fórmula:** En estas fórmulas la notación “máx” representa “el máximo de todos”. El desplazamiento para cada dimensión geográfica se representa como `dimensiónOffset` (por ejemplo, `xOffset` es el valor de desplazamiento aplicado a todas las coordenadas X). El factor de escala para cada dimensión geográfica se representa como `dimensiónScale` (por ejemplo, `xScale` es el factor de escala aplicado a todas las coordenadas X).

```
(max(x) - xOffset) * xScale = 253  
(max(y) - yOffset) * yScale = 253  
(max(z) - zoffset) * zScale = 253  
(max(m) - moffset) * mScale = 253
```

Cuando seleccione qué factores de escala funcionan mejor con sus datos de coordenadas, asegúrese de:

- Utilizar el mismo factor de escala para las coordenadas X e Y.
- Que cuando se multiplique por una coordenada decimal X o por una coordenada decimal Y, el factor de escala dé como resultado un valor menor que  $2^{53}$ . Una técnica habitual consiste en convertir el factor de escala en una potencia de 10. Es decir, el factor de escala deberá ser 10 a la primera potencia (10), 10 a la segunda potencia (100), 10 a la tercera potencia (1000) o, si es necesario, un factor superior.
- Que el factor de escala sea lo suficientemente grande como para garantizar que el número de dígitos del nuevo número entero sea el mismo que el de dígitos significantes de la coordenada decimal original.

## Ejemplo

Supongamos que la entrada de la función `ST_Point` consta de una coordenada X de 10.01 y una coordenada Y de 20.03, además del identificador de un sistema de referencia espacial. Cuando se invoca a `ST_Point`, la función multiplica el valor 10.01 y el valor 20.03 por el factor de escala del sistema de referencia espacial para las coordenadas X e Y. Si este factor de escala es 10, los números enteros resultantes que almacenará `Spatial Extender` serán 100 y 200, respectivamente. Debido a que el número de dígitos significantes de estos enteros (3) es menor que el número de dígitos significantes de las coordenadas (4), `Spatial Extender` no podrá revertir estos enteros a las coordenadas originales ni obtener a partir de ellos valores que sean coherentes con el sistema de coordenadas al que pertenecen dichas coordenadas. Sin embargo, si el factor de escala es 100, los números enteros resultantes que `DB2 Spatial Extender` almacena serán 1001 y 2003: valores que se

pueden volver a convertir a las coordenadas originales o a partir de los cuales se pueden derivar coordenadas compatibles.

### **Unidades para valores de desplazamiento y factores de escala**

Tanto si utiliza un sistema de referencia espacial existente como si crea uno nuevo, las unidades para los valores de desplazamiento y los factores de escala variarán según el tipo de sistema de coordenadas que utilice. Por ejemplo, si utiliza un sistema de coordenadas geográficas, los valores estarán en unidades angulares como por ejemplo, los grados decimales; si utiliza un sistema de coordenadas proyectadas, los valores estarán en unidades lineales, como los metros o los pies.

## **Creación de un sistema de referencia espacial**

Cree un sistema de referencia espacial nuevo si ninguno de los sistemas de referencia espacial proporcionados con DB2 Spatial Extender funcionan con los datos.

Para realizar esta tarea:

1. Elija la interfaz.

Puede crear un sistema de referencia espacial de cualquiera de las maneras siguientes:

- Utilice la ventana Crear sistema de referencia espacial del Centro de control de DB2. Para obtener más información sobre cómo utilizar esta ventana, consulte la ayuda en línea.
- Emita el mandato `db2se create_srs` desde el procesador de línea de mandatos de `db2se`.
- Ejecute una aplicación que invoque al procedimiento almacenado `db2se.ST_create_srs`.

2. Especifique un ID de sistema de referencia espacial (SRID) apropiado.

- Para los datos geodésicos en una representación terrestre redonda, especifique un valor de SRID comprendido entre 200000318 y 2000001000.
- Para los datos espaciales en una representación terrestre plana, especifique un SRID que no esté todavía definido.

3. Decida el grado de precisión que desea.

Tiene un de estas dos opciones:

- Especificar la extensión de la zona geográfica con la que esté trabajando y los factores de escala que desee utilizar con los datos de coordenadas. Spatial Extender toma las extensiones que haya especificado y calcula el desplazamiento. Puede especificar extensiones de una de las maneras siguientes:
  - Elija **Extensiones** en la ventana Crear sistema de referencia espacial del Centro de control.
  - Proporcione los parámetros apropiados para el mandato `db2se create_srs` o el procedimiento almacenado `db2se.ST_create_srs`.
- Especificar tanto los valores de desplazamiento (necesarios para que Spatial Extender convierta valores negativos en valores positivos) como los factores de escala (necesarios para que Spatial Extender convierta valores decimales en números enteros). Utilice este método cuando necesite seguir criterios estrictos para lograr mayor precisión. Puede especificar valores de desplazamiento y factores de escala de una de las maneras siguientes:
  - Elija **Desplazamiento** en la ventana Crear sistema de referencia espacial del Centro de control

- Proporcione los parámetros apropiados para el mandato `db2se create_srs` o el procedimiento almacenado `db2se.ST_create_srs`.
4. Calcule la información de conversión que Spatial Extender necesita para convertir datos de coordenadas en enteros positivos, y proporcione esta información a la interfaz que haya elegido.

Esta información será distinta según el método que elija en el paso 3.

- Si en el paso 3 eligió el método Extensiones, deberá calcular la información siguiente:
  - Factores de escala. Si alguna de las coordenadas con las que trabaja son valores decimales, calcule factores de escala. Los factores de escala son números que, multiplicados por medidas y coordenadas decimales, dan lugar a números enteros que tienen como mínimo el mismo número de dígitos significantes que las medidas y coordenadas originales. Si las coordenadas son valores enteros, los factores de escala pueden definirse en 1. Si las coordenadas son valores decimales, el factor de escala debe definirse en un número que convierta la parte decimal en un valor entero. Por ejemplo, si las unidades de coordenadas son metros y la exactitud de los datos es 1 cm., necesitará un factor de escala de 100.
  - Valores mínimos y máximos para las coordenadas y las medidas.
- Si en el paso 3, eligió el método Desplazamiento, deberá calcular la información siguiente:
  - Valores de desplazamiento
 

Si los datos de coordenadas incluyen números o medidas negativos, deberá especificar los valores de desplazamiento que desee utilizar. Un desplazamiento es un número que se resta de todas las coordenadas, dejando sólo valores positivos como resto. Si trabaja con coordenadas positivas, establezca todos los valores de desplazamiento en 0. Si no trabaja con coordenadas positivas, seleccione un desplazamiento que, al aplicarse a los datos de coordenadas, dé como resultado números enteros que sean menores que el valor del mayor número entero positivo (9,007,199,254,740,992).
  - Factores de escala
 

Si alguna de las coordenadas para las ubicaciones que está representando es un número decimal, determine qué factores de escala se deben utilizar y entre estos factores de escala en la ventana Crear sistema de referencia espacial.

5. Envíe el mandato `db2se create_srs` o el procedimiento almacenado `db2se.ST_create_srs` para crear el procedimiento almacenado.
- Por ejemplo, el siguiente mandato crear un sistema de referencia espacial denominado `mysrs`:

```
db2se create_srs mydb -srsName \"mysrs\"
-srsID 100 -xScale 10 -coordsysName
\"GCS_North_American_1983\"
```

## Cálculo de los factores de escala

### Requisitos previos

Si crea un sistema de referencia espacial y cualquiera de las coordenadas con las que está trabajando son valores decimales, calcule los factores de escala apropiados para las coordenadas y las medidas utilizadas. Los factores de escala son números

que, multiplicados por medidas y coordenadas decimales, dan lugar a números enteros que tienen como mínimo el mismo número de dígitos significantes que las medidas y coordenadas originales.

Después de calcular los factores de escala deberá determinar los valores de extensión. A continuación, emita el mandato `db2se create_srs` o el procedimiento almacenado `db2se.ST_create_srs`.

Para calcular los factores de escala:

1. Determina qué coordenadas X e Y son números decimales o tienen bastante probabilidad de serlo. Por ejemplo, supongamos que determina que tres de las diversas coordenadas X e Y que va a manejar son números decimales: 1,23, 5,1235 y 6,789.
2. Busque la coordenada decimal que tenga la precisión decimal más larga. A continuación, determine por qué potencia de 10 se puede multiplicar esta coordenada para obtener un número entero de igual precisión. Por ejemplo, de las tres coordenadas decimales del ejemplo actual, 5,1235 tiene la precisión decimal más larga. Si la multiplicamos por 10 a la potencia cuatro (10000), obtendremos el número entero 51235.
3. Determine si el entero generado por la multiplicación que se acaba de describir es menor que  $2^{53}$ . 51235 no es una cifra demasiado alta. Pero supongamos que, además de 1,23, 5,11235 y 6,789, su rango de coordenadas X e Y incluye un cuarto valor decimal, 10000000006,789876. Puesto que la precisión decimal de la coordenada es más larga que la de las otras tres, tendrá que multiplicar esta coordenada y no 5,1235 por una potencia de 10. Para convertirla en un número entero, la tendría que multiplicar por 10 a la sexta potencia (1000000). Sin embargo, el valor resultante, 10000000006789876, es mayor que  $2^{53}$ . Si DB2 Spatial Extender intentara almacenarlo, los resultados serían imprevisibles.

Para evitar este problema, seleccione una potencia de 10 que, multiplicada por la coordenada original, dé como resultado un número decimal que DB2 Spatial Extender pueda truncar en un número entero susceptible de almacenamiento, con una pérdida de precisión mínima. En este caso, podría seleccionar 10 a la potencia cinco (100000). Multiplicando 100000 por 10000000006.789876 obtendrá 1000000000678987.6. DB2 Spatial Extender redondearía este número a 1000000000678988, reduciendo ligeramente su precisión.

## Factores de conversión que transforman datos de coordenadas en enteros

DB2 Spatial Extender utiliza valores de desplazamiento y factores de escala para convertir los datos de coordenadas proporcionados en enteros positivos. Los sistemas de referencia espacial por omisión ya tienen valores de desplazamiento y factores de escala asociados a los mismos. Si crea un nuevo sistema de referencia espacial, determine los factores de escala y, opcionalmente, los valores de desplazamiento que funcionen mejor con sus datos.

## Determinación de las coordenadas y medidas mínimas y máximas

Utilice este procedimiento para determinar coordenadas y medidas mínimas y máximas si:



- Decide crear un nuevo sistema de referencia espacial nuevo porque ninguno de los sistemas de referencia espacial proporcionados con DB2 Spatial Extender funcionan con los datos.
- Decide utilizar transformaciones de extensión para convertir las coordenadas.

Determine las coordenadas y medidas máximas y mínimas si decide especificar transformaciones de extensión cuando cree un sistema de referencia espacial.

Una vez determinados los valores de extensión, si alguna de las coordenadas son valores decimales, deberá calcular los factores de escala. De lo contrario, emita el mandato `db2se create_srs` o el procedimiento almacenado `db2se.ST_create_srs`.

Para determinar las coordenadas y medidas mínimas y máximas de las ubicaciones que desee representar:

1. Determine las coordenadas X mínimas y máximas. Para hallar la coordenada X mínima, identifique la coordenada X de su dominio que se encuentre más al oeste. (Si la ubicación se encuentra al oeste del punto de origen, esta coordenada será un valor negativo). Para hallar la coordenada X máxima, identifique la coordenada X de su dominio que se encuentre más al este. Por ejemplo, si está representando pozos de petróleo y cada uno de ellos está definido por un par de coordenadas X e Y, la coordenada X que indica la ubicación del pozo de petróleo que está situado más al oeste es la coordenada X mínima y la coordenada X que indica la ubicación del pozo de petróleo situado más al este es la coordenada X máxima.

**Consejo:** Para los tipos formados por varios elementos, como los multipolígonos, asegúrese de seleccionar el punto más lejano del polígono más lejano en la dirección que esté calculando. Por ejemplo, si intenta identificar la coordenada X mínima, identifique la coordenada X que se encuentre más al oeste del polígono que esté más al oeste en el multipolígono.

2. Determine las coordenadas Y mínimas y máximas. Para localizar la coordenada Y mínima, identifique la coordenada Y de su dominio que se encuentre más al sur. (Si la ubicación se encuentra al sur del punto de origen, esta coordenada será un valor negativo). Para determinar la coordenada Y máxima, identifique la coordenada Y de su dominio que se encuentre más al sur.
3. Determine las coordenadas Z mínimas y máximas. La coordenada Z mínima es la coordenada de profundidad mayor y la coordenada Z máxima es la coordenada de altura mayor.
4. Determine las medidas mínimas y máximas. Si va a incluir medidas en los datos espaciales, determine qué medida tiene el valor numérico más alto y qué medida tiene el más bajo.

## Cálculo de los valores de desplazamiento

Si los datos de coordenadas incluyen números o medidas negativos, deberá especificar valores de desplazamiento.

Si crea un sistema de referencia espacial y los datos de coordenadas incluyen números o medidas negativos, deberá especificar los valores de desplazamiento que desea utilizar. Un desplazamiento es un número que se resta de todas las coordenadas, dejando sólo valores positivos como resto. Puede mejorar el rendimiento de las operaciones espaciales si las coordenadas son enteros positivos en vez de números o medidas negativas.

Para calcular los valores de desplazamiento para las coordenadas con las que esté trabajando:

1. Determine las coordenadas negativas X, Y y Z más bajas dentro del rango de coordenadas para las ubicaciones que desea representar. Si los datos van a incluir medidas negativas, determine las más bajas de estas medidas.
2. Opcional pero recomendado: Indique a DB2 Spatial Extender que el dominio que abarca las ubicaciones que le interesan es mayor de lo que es en realidad. En consecuencia, después de escribir datos sobre estas ubicaciones en una columna espacial, puede añadir datos sobre ubicaciones de nuevos elementos conforme estos se añaden a los bordes externos del dominio, sin necesidad de sustituir el sistema de referencia espacial por otro.

Para cada coordenada y medida que haya identificado en el paso 1, añada una cantidad equivalente entre el cinco y el diez por ciento de la coordenada o medida. El resultado se conoce como un *valor aumentado*. Por ejemplo, si la coordenada X negativa más baja es -100, podría sumarle -5, obteniendo como resultado un valor aumentado de -105. Posteriormente, cuando cree el sistema de referencia espacial, indicará que la coordenada X más baja es -105, en lugar del valor real de -100. DB2 Spatial Extender interpretará entonces que -105 es el límite situado más al oeste del dominio.

3. Encuentre un valor que, al restarlo del valor X aumentado, dé como resultado cero; será el valor de desplazamiento para las coordenadas X. DB2 Spatial Extender resta este número de todas las coordenadas X para producir solo valores positivos.

Por ejemplo, si el valor X aumentado es -105, deberá restarle -105 para obtener 0. DB2 Spatial Extender restará entonces -105 de todas las coordenadas X asociadas con los elementos que se estén representando. Puesto que ninguna de estas coordenadas es mayor que -100, todos los valores resultantes de la sustracción serán positivos.

4. Repita el paso 3 para el valor Y aumentado, el valor Z aumentado y la medida aumentada.

## Creación de un sistema de referencia espacial

Cree un sistema de referencia espacial nuevo si ninguno de los sistemas de referencia espacial proporcionados con DB2 Spatial Extender funcionan con los datos.

Para realizar esta tarea:

1. Elija la interfaz.  
Puede crear un sistema de referencia espacial de cualquiera de las maneras siguientes:
  - Utilice la ventana Crear sistema de referencia espacial del Centro de control de DB2. Para obtener más información sobre cómo utilizar esta ventana, consulte la ayuda en línea.
  - Emita el mandato `db2se create_srs` desde el procesador de línea de mandatos de `db2se`.
  - Ejecute una aplicación que invoque al procedimiento almacenado `db2se.ST_create_srs`.
2. Especifique un ID de sistema de referencia espacial (SRID) apropiado.
  - Para los datos geodésicos en una representación terrestre redonda, especifique un valor de SRID comprendido entre 200000318 y 2000001000.
  - Para los datos espaciales en una representación terrestre plana, especifique un SRID que no esté todavía definido.

3. Decida el grado de precisión que desea.

Tiene un de estas dos opciones:

- Especificar la extensión de la zona geográfica con la que esté trabajando y los factores de escala que desee utilizar con los datos de coordenadas. Spatial Extender toma las extensiones que haya especificado y calcula el desplazamiento. Puede especificar extensiones de una de las maneras siguientes:
    - Elija **Extensiones** en la ventana Crear sistema de referencia espacial del Centro de control.
    - Proporcione los parámetros apropiados para el mandato `db2se create_srs` o el procedimiento almacenado `db2se.ST_create_srs`.
  - Especificar tanto los valores de desplazamiento (necesarios para que Spatial Extender convierta valores negativos en valores positivos) como los factores de escala (necesarios para que Spatial Extender convierta valores decimales en números enteros). Utilice este método cuando necesite seguir criterios estrictos para lograr mayor precisión. Puede especificar valores de desplazamiento y factores de escala de una de las maneras siguientes:
    - Elija **Desplazamiento** en la ventana Crear sistema de referencia espacial del Centro de control
    - Proporcione los parámetros apropiados para el mandato `db2se create_srs` o el procedimiento almacenado `db2se.ST_create_srs`.
4. Calcule la información de conversión que Spatial Extender necesita para convertir datos de coordenadas en enteros positivos, y proporcione esta información a la interfaz que haya elegido.

Esta información será distinta según el método que elija en el paso 3.

- Si en el paso 3 eligió el método Extensiones, deberá calcular la información siguiente:
  - Factores de escala. Si alguna de las coordenadas con las que trabaja son valores decimales, calcule factores de escala. Los factores de escala son números que, multiplicados por medidas y coordenadas decimales, dan lugar a números enteros que tienen como mínimo el mismo número de dígitos significantes que las medidas y coordenadas originales. Si las coordenadas son valores enteros, los factores de escala pueden definirse en 1. Si las coordenadas son valores decimales, el factor de escala debe definirse en un número que convierta la parte decimal en un valor entero. Por ejemplo, si las unidades de coordenadas son metros y la exactitud de los datos es 1 cm., necesitará un factor de escala de 100.
  - Valores mínimos y máximos para las coordenadas y las medidas.
- Si en el paso 3, eligió el método Desplazamiento, deberá calcular la información siguiente:
  - Valores de desplazamiento  
Si los datos de coordenadas incluyen números o medidas negativos, deberá especificar los valores de desplazamiento que desee utilizar. Un desplazamiento es un número que se resta de todas las coordenadas, dejando sólo valores positivos como resto. Si trabaja con coordenadas positivas, establezca todos los valores de desplazamiento en 0. Si no trabaja con coordenadas positivas, seleccione un desplazamiento que, al aplicarse a los datos de coordenadas, dé como resultado números enteros que sean menores que el valor del mayor número entero positivo (9,007,199,254,740,992).
  - Factores de escala

Si alguna de las coordenadas para las ubicaciones que está representando es un número decimal, determine qué factores de escala se deben utilizar y entre estos factores de escala en la ventana Crear sistema de referencia espacial.

5. Envíe el mandato `db2se create_srs` o el procedimiento almacenado `db2se.ST_create_srs` para crear el procedimiento almacenado.

Por ejemplo, el siguiente mandato crear un sistema de referencia espacial denominado `mysrs`:

```
db2se create_srs mydb -srsName \"mysrs\"  
-srsID 100 -xScale 10 -coordsysName  
\"GCS_North_American_1983\"
```

---

## Capítulo 9. Configuración de columnas espaciales

En la preparación para obtener datos espaciales para un proyecto, no sólo selecciona o crea un sistema de coordenadas y un sistema de referencia espacial; también proporciona una o varias columnas de tabla donde contener los datos.

Este capítulo:

- Indica que los resultados de las consultas de las columnas se pueden obtener gráficamente y proporciona directrices para seleccionar tipos de datos para las columnas
- Describe la tarea de proporcionar las columnas
- Describe la tarea de hacer las columnas accesibles a herramientas que puedan visualizar su contenido en forma gráfica

---

### Columnas espaciales

#### Columnas espaciales con contenido visualizable

Cuando utiliza una herramienta de visualización, como ArcExplorer for DB2<sup>®</sup>, para consultar una columna espacial, la herramienta devuelve resultados en forma de una representación geográfica; por ejemplo, un mapa de límites de parcelas o el esquema de un sistema de carreteras. Algunas herramientas de visualización requieren que todas las filas de la columna utilicen el mismo sistema de referencia espacial. Para aplicar esta restricción, debe registrar la columna en un sistema de referencia espacial.

#### Tipos de datos espaciales

Cuando habilita una base de datos para operaciones espaciales, DB2 Spatial Extender proporciona a la base de datos una jerarquía de tipos de datos estructurados.

La Figura 12 en la página 60 presenta esta jerarquía. En esta figura, los tipos de los que pueden crearse instancias tienen un fondo blanco; los tipos de los que no pueden crearse instancias tienen un fondo gris.

Los tipos de datos de los que se pueden crear instancias son ST\_Point, ST\_LineString, ST\_Polygon, ST\_GeomCollection, ST\_MultiPoint, ST\_MultiPolygon y ST\_MultiLineString.

Los tipos de datos de los que no se pueden crear instancias son ST\_Geometry, ST\_Curve, ST\_Surface, ST\_MultiSurface y ST\_MultiCurve.

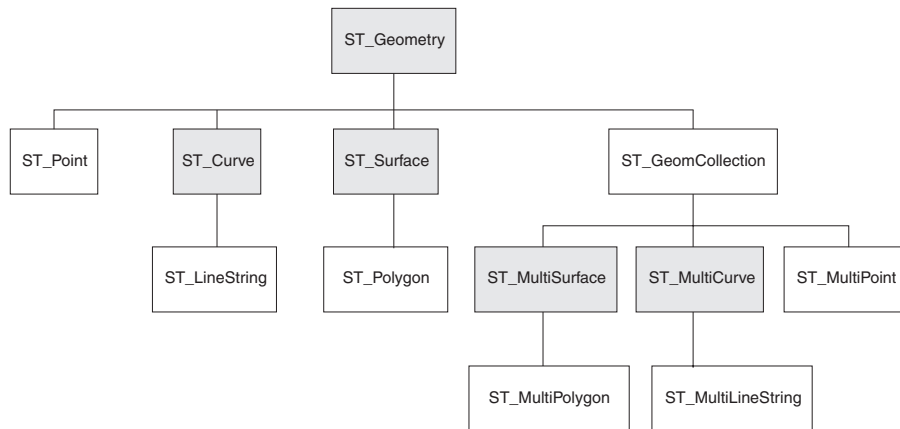


Figura 12. Jerarquía de tipos de datos espaciales. De los tipos de datos que aparecen en los cuadros blancos pueden crearse instancias. De los tipos de datos que aparecen en los cuadros sombreados no pueden crearse instancias.

La jerarquía de la Figura 12 incluye:

- Tipos de datos para elementos geográficos que se pueden percibir como formados por una sola unidad; por ejemplo, edificios individuales y lagos aislados.
- Tipos de datos para elementos geográficos que constan de varias unidades o componentes; por ejemplo, sistemas de canales o grupos de islas en un lago.
- Un tipo de datos para elementos geográficos de todo tipo.

### Tipos de datos para características de una sola unidad

Utilice `ST_Point`, `ST_LineString` y `ST_Polygon` para almacenar coordenadas que definen el espacio ocupado por elementos que se pueden percibir como una sola unidad.

- Utilice `ST_Point` cuando desee indicar el punto en el espacio ocupado por un elemento geográfico aislado. El elemento puede ser muy pequeño (un pozo de agua), muy grande (una ciudad) o de tamaño intermedio (un complejo de edificios o un parque). En cada caso, el punto en el espacio se puede situar en la intersección de una línea de coordenadas este-oeste (por ejemplo, un paralelo) y una línea de coordenadas norte-sur (por ejemplo, un meridiano). Un dato `ST_Point` incluye una coordenada X y una coordenada Y que definen una intersección de este tipo. La coordenada X indica el lugar donde se encuentra la intersección en la línea este-oeste; la coordenada Y indica el lugar donde se encuentra la intersección en la línea norte-sur.
- Utilice `ST_LineString` para coordenadas que definan el espacio ocupado por elementos lineales; por ejemplo, calles, canales y conductos.
- Utilice `ST_Polygon` cuando desee indicar la extensión de espacio ocupada por un elemento que conste de varios lados; por ejemplo, un barrio de una ciudad, un bosque o un hábitat natural. Un dato de `ST_Polygon` consta de las coordenadas que definen el perímetro de un elemento de este tipo.

En algunos casos, se puede utilizar `ST_Polygon` y `ST_Point` para un mismo elemento. Por ejemplo, supongamos que necesita información espacial sobre un complejo de apartamentos. Si desea representar el punto en el espacio en el que se encuentra cada edificio en el complejo, utilizaría `ST_Point` para guardar las coordenadas X e Y que definen ese punto. De lo contrario, si desea representar el área ocupada por todo el complejo, debe utilizar `ST_Polygon` para almacenar las coordenadas que definen el perímetro de esta zona.

## Tipos de datos para elementos geográficos formados por una sola unidad

Utilice `ST_MultiPoint`, `ST_MultiLineString` y `ST_MultiPolygon` para almacenar coordenadas que definan espacios ocupados por elementos geográficos formados por varias unidades.

- Utilice `ST_MultiPoint` para representar elementos geográficos formados de unidades cuyas ubicaciones estén referenciadas por una coordenada X y una coordenada Y. Por ejemplo, considere una tabla cuyas filas representan cadenas de islas. Se han identificado la coordenada X y la coordenada Y para cada isla. Si desea que la tabla incluya estas coordenadas y las coordenadas para cada cadena en su totalidad, defina una columna `ST_MultiPoint` para albergar estas coordenadas.
- Utilice `ST_MultiLineString` cuando esté representando elementos compuestos de unidades lineales y desee almacenar las coordenadas para las ubicaciones de estas unidades y la ubicación para cada elemento en su totalidad. Por ejemplo, imagine una tabla cuyas filas representen sistemas fluviales. Si desea que la tabla incluya coordenadas para las ubicaciones de los sistemas y los componentes de los mismos, defina una columna `ST_MultiLineString` para albergar estas coordenadas.
- Utilice `ST_MultiPolygon` cuando esté representando elementos compuestos de unidades que tengan varios lados y desee almacenar las coordenadas para las ubicaciones de estas unidades y la ubicación de cada elemento en su totalidad. Por ejemplo, imagine una tabla cuyas filas representen regiones rurales o las granjas en cada región. Si desea que la tabla incluya coordenadas para las ubicaciones de las regiones y las granjas, defina una columna `ST_MultiPolygon` para albergar estas coordenadas.

Las múltiples unidades que forman el elemento geográfico no son una serie de entidades individuales, sino que son un agregado de las partes que forma la entidad completa.

## Un tipo de datos para todos los elementos geográficos

Puede utilizar `ST_Geometry` cuando no esté seguro de cuál de los otros tipos de datos debe utilizar.

Puesto que `ST_Geometry` es la raíz de la jerarquía a la que pertenecen los demás tipos de datos, una columna `ST_Geometry` puede contener los mismos tipos de datos que se pueden almacenar en columnas de los demás tipos de datos.

**Atención:** Si piensa utilizar el geocodificador proporcionado, `DB2SE_USA_GEOCODER`, para crear datos para una columna espacial, la columna debe ser de tipo `ST_Point` o `ST_Geometry`. Sin embargo, determinadas herramientas de visualización no dan soporte a columnas `ST_Geometry`, sino sólo a columnas a las que se ha asignado un subtipo apropiado de `ST_Geometry`.

---

## Creación de columnas espaciales

Debe crear columnas espaciales para almacenar y recuperar datos espaciales.

Antes de crear una columna espacial, su ID de usuario debe tener las autorizaciones necesarias sobre las sentencias SQL `CREATE TABLE` o `ALTER TABLE` de DB2. El ID de usuario debe tener como mínimo uno de los privilegios o autorizaciones siguientes:

- `DBADM` sobre la base de datos donde reside la tabla que contiene la columna

- Autorización CREATETAB sobre la base de datos y privilegio USE para el espacio de tabla, así como uno de los siguientes:
  - Autorización IMPLICIT\_SCHEMA sobre la base de datos, si el esquema del índice no existe
  - Privilegio CREATEIN sobre el esquema, si el nombre de esquema del índice hace referencia a un esquema existente
- Privilegio ALTER para la tabla que se debe modificar
- Privilegio CONTROL para la tabla que se debe modificar
- Privilegio ALTERIN para el esquema de la tabla que debe alterarse

Esta tarea forma parte de la tarea más amplia "Configurar recursos espaciales para un proyecto." Después de elegir un sistema de coordenadas y determinar qué sistema de referencia espacial utilizar para los datos, cree una columna espacial en una tabla existente o importe datos espaciales a una tabla nueva.

Para realizar esta tarea... :

Puede proporcionar columnas espaciales a la base de datos de una o de varias maneras:

- Utilice la sentencia CREATE TABLE de DB2 para crear una tabla y para incluir una columna espacial en la tabla.
- Utilice la sentencia ALTER TABLE de DB2 para añadir una columna espacial a una tabla existente.
- Utilice la ventana Crear columna espacial del Centro de control de DB2. Abra la ventana Columnas espaciales desde una tabla.
- Si está importando datos espaciales desde un archivo de formas, utilice DB2 Spatial Extender para crear una tabla y para proporcionar a esta tabla una columna donde alojar los datos. Consulte "Importación de datos de formas a una tabla nueva o existente".

La siguiente tarea para configurar recursos espaciales es "Registrar columnas espaciales."

---

## Registro de columnas espaciales

Si se registra una columna espacial, se crea una restricción en la tabla, si es posible, para asegurar que todas las geometrías utilicen el sistema de referencia espacial especificado.

Requisitos previos

Es posible que desee registrar una columna espacial en los siguientes casos:

- Acceso mediante las herramientas de visualización
 

Si desea que determinadas herramientas de visualización (por ejemplo, ArcExplorer for DB2) generen visualizaciones gráficas de los datos en una columna espacial, deberá asegurarse de la integridad de los datos de la columna. Esto se hace imponiendo una restricción que requiere que todas las filas de la columna utilicen el mismo sistema de referencia espacial. Para imponer esta restricción, registre la columna, especificando su nombre y el sistema de referencia espacial que se aplica a la misma.
- Acceso mediante índices espaciales
 

Utilice el mismo sistema de coordenadas para todos los datos de una columna espacial en la que desee crear un índice para asegurarse de que el índice espacial



devuelva los resultados correctos. La columna espacial se registra para forzar que los datos utilicen el mismo sistema de referencia espacial y, como corresponde, el mismo sistema de coordenadas.



---

## Capítulo 10. Cómo llenar columnas espaciales

Después de crear columnas espaciales, y registrar las que se deberán acceder mediante herramientas de visualización, está preparado para llenar las columnas con datos espaciales. Existen tres maneras de proporcionar los datos: importarlos; utilizar un geocodificador para obtenerlos de los datos comerciales; o bien utilizar funciones espaciales para crearlos o para obtenerlos de los datos comerciales o de otros datos espaciales.

---

### Acerca de la importación y de la exportación de datos espaciales

Puede utilizar DB2<sup>®</sup> Spatial Extender para intercambiar datos espaciales entre la base de datos y las fuentes de datos externas. Más concretamente, puede importar datos espaciales desde fuentes externas transfiriéndolos a la base de datos en archivos, denominados archivos de intercambio de datos. También puede exportar datos espaciales desde la base de datos a archivos de intercambio de datos desde donde fuentes externas pueden adquirirlos. Esta sección sugiere algunas de las razones para importar y exportar datos espaciales y describe la naturaleza de los archivos de intercambio de datos que DB2 Spatial Extender soporta.

#### Razones para importar y exportar datos espaciales

Importando datos espaciales, puede obtener mucha información espacial que ya está disponible en la empresa. Exportándolos, puede hacer que estén disponibles en un formato de archivo estándar para aplicaciones existentes. Supongamos los casos siguientes:

- La base de datos contiene datos espaciales que representan oficinas de ventas, clientes y otros elementos de su empresa. Desea complementar estos datos con datos espaciales que representen el entorno cultural de su organización: ciudades, calles, puntos de interés, etc. Los datos que desea pueden obtenerse de un proveedor de mapas. Puede utilizar DB2 Spatial Extender para importarlos desde un archivo de intercambio de datos que suministra el proveedor.
- Desea migrar los datos espaciales desde un sistema Oracle al entorno DB2. Puede continuar utilizando el programa de utilidad de Oracle para grabar datos en un archivo de intercambio de datos. Luego utiliza DB2 Spatial Extender para importar los datos de este archivo a la base de datos que tiene habilitada para operaciones espaciales.
- No está conectado a DB2 y desea utilizar un geonavegador para mostrar presentaciones visuales de información espacial a los clientes. El navegador sólo necesita archivos con los que trabajar; no necesita estar conectado a una base de datos. Podría utilizar DB2 Spatial Extender para exportar los datos a un archivo de intercambio de datos y luego utilizar un navegador para producir los datos en formato visual.

#### Archivos de formas

DB2 Spatial Extender soporta archivos de formas para el intercambio de datos. En realidad, el término archivos de formas designa un conjunto de archivos que tienen el mismo nombre de archivo pero diferentes extensiones de archivo. Este grupo puede incluir hasta cuatro archivos. Estas funciones son:

- Un archivo que contiene datos espaciales en formato de forma, un formato estándar de facto de la industria desarrollado por ESRI. Estos datos se denominan a menudo datos de formas. La extensión de un archivo que contiene datos de formas es .shp.
- Un archivo que contiene datos comerciales pertenecientes a ubicaciones definidas por datos de formas. La extensión de este archivo es .dbf.
- Un archivo que contiene un índice para datos de formas. La extensión de este archivo es .shx.
- Un archivo que contiene una especificación del sistema de coordenadas en el que se basan los datos en un archivo .shp. La extensión de este archivo es .prj.

Los archivos de formas se suelen utilizar para importar datos procedentes de sistemas de archivos y para exportar datos a archivos en sistemas de archivos.

Cuando utiliza DB2 Spatial Extender para importar datos de formas, recibe como mínimo un archivo .shp. En la mayoría de los casos, recibirá también uno o más de los otros tres tipos de archivos de formas.

---

## Importación de datos espaciales

Esta sección proporciona una visión general de las tareas de importar archivos de formas y archivos de transferencia SDE a la base de datos. La sección incluye referencias cruzadas a información detallada que necesita saber (por ejemplo, procesos y parámetros) para poder realizar estas tareas.

### Importación de datos de formas a una tabla nueva o existente

Puede importar datos de formas a una tabla o vista existente, o puede crear una tabla e importar los datos de formas a dicha tabla en una sola operación.

Antes de importar datos de formas a una tabla o vista existente, el ID de usuario debe tener una de las siguientes formas de autorización o privilegios:

- Autorización DATAACCESS sobre la base de datos que contiene la tabla o la vista
- Privilegio CONTROL sobre la tabla o la vista
- Privilegio INSERT sobre la tabla o la vista
- Privilegio SELECT sobre la tabla o la vista (necesario sólo si la tabla incluye una columna de ID que no es una columna IDENTITY)

Además debe tener uno de los siguientes privilegios:

- Privilegios para acceder a los directorios a los que pertenecen los archivos de entrada y los archivos de error
- Privilegios de lectura sobre los archivos de entrada y privilegios de escritura sobre los archivos de error

Antes de empezar a crear automáticamente una tabla e importar los datos de formas a la misma, el ID de usuario debe tener las siguientes formas de autorización o privilegios:

- Autorización DBADM sobre la base de datos que contiene la tabla o la vista
- Autorización CREATETAB sobre la base de datos que contiene la tabla o la vista
- Si el esquema ya existe, privilegio CREATEIN sobre el esquema al que pertenece la tabla

- Si el esquema especificado para la tabla no existe, autorización IMPLICIT\_SCHEMA sobre la base de datos que contiene la tabla

Además debe tener uno de los siguientes privilegios:

- Privilegios para acceder a los directorios a los que pertenecen los archivos de entrada y los archivos de error
- Privilegios de lectura sobre los archivos de entrada y privilegios de escritura sobre los archivos de error

Elija uno de los métodos siguientes para importar datos de forma:

- Importar los datos de forma a una columna espacial en una tabla existente, una vista actualizable existente o una vista existente en la que está definido un desencadenante INSTEAD OF para las operaciones INSERT.
- Crear automáticamente una tabla con una columna espacial e importar los datos de formas a esta columna.

Para realizar esta tarea:

Elija qué método desea utilizar para importar datos de formas:

- Utilice la ventana Importar datos de formas del Centro de control de DB2.
- Emita el mandato `db2se import_shape`.
- Ejecute una aplicación que llame al procedimiento almacenado `db2gse.ST_import_shape`.

---

## Exportación de datos espaciales

Esta sección proporciona una visión general de las tareas de exportar datos espaciales a archivos de formas y de transferencia SDE. La sección incluye referencias cruzadas a información detallada que necesita saber (por ejemplo, procesos y parámetros) para poder realizar estas tareas.

### Exportación de datos a un archivo de formas

Puede exportar los datos espaciales devueltos en los resultados de una consulta a un archivo de formas.

Para poder exportar datos a un archivo de formas, su ID de usuario debe tener los privilegios siguientes:

- Privilegio para ejecutar una subselección que devuelve los resultados que desea exportar
- Privilegio para escribir en el directorio donde reside el archivo al que exportará los datos
- Privilegio para crear un archivo donde colocar los datos exportados (necesario si dicho archivo no existe aún)

Para saber qué son estos privilegios y cómo obtenerlos, consulte con el administrador de la base de datos.

Los datos espaciales que se exportan a un archivo de formas pueden proceder de fuentes tales como una tabla base, una unión de varias tablas, conjuntos de resultados devueltos al consultar vistas o datos de salida de una función espacial.

Si existe un archivo al que desea exportar datos, DB2 Spatial Extender puede añadir los datos a ese archivo. Si dicho archivo no existe, puede utilizar DB2 Spatial Extender para crear uno.

Para realizar esta tarea:

Elija un método para exportar datos a un archivo de formas:

- Inicialice la exportación desde la ventana Exportar archivo de formas del Centro de control de DB2.
- Emita el mandato `db2se` desde el procesador de línea de mandatos `db2se`.
- Ejecute una aplicación que llame al procedimiento almacenado `db2gse.ST_export_shape`.

---

## Cómo utilizar un geocodificador

Esta sección trata acerca del concepto de la geocodificación y presenta las tareas siguientes:

- Definición del trabajo que desea que realice un geocodificador; por ejemplo, especificación de cuántos registros debe procesar el geocodificador antes de enviar una confirmación
- Configuración de un geocodificador para geocodificar datos tan pronto como se añadan los datos a la tabla o se actualicen en la misma.
- Ejecución de un geocodificador en modalidad de proceso por lotes

## Geocodificadores y geocodificación

Los términos geocodificador y geocodificación se utilizan en varios contextos. Esta explicación clasifica estos contextos, de forma que los significados de los términos queden claros cada vez que se encuentre con uno de ellos. La explicación define geocodificador y geocodificación, describe las modalidades en las que funciona un geocodificador, describe una actividad más amplia a la que pertenece la geocodificación, y resume las tareas de los usuarios que pertenecen a la geocodificación.

En DB2® Spatial Extender, un geocodificador es una función escalar que convierte los datos existentes (la entrada de la función) en datos que se pueden comprender en términos espaciales (la salida de la función). Normalmente, los datos existentes son datos relacionales que describen o nombran una ubicación. Por ejemplo, el geocodificador que se proporciona con DB2 Spatial Extender, `DB2SE_USA_GEOCODER`, convierte direcciones de Estados Unidos a datos `ST_Point`. DB2 Spatial Extender también puede dar soporte a geocodificadores proporcionados por proveedores y a geocodificadores proporcionados por usuarios; y los datos de entrada y de salida de los mismos puede que no sean los mismos que con `DB2SE_USA_GEOCODER`. Para poner un ejemplo: Es posible que un geocodificador proporcionado por un proveedor convierta las direcciones en coordenadas que DB2 no almacene, sino que las grabe en un archivo. Es posible que otro pueda convertir el número de una oficina en un edificio comercial a coordenadas que definan la ubicación de la oficina en el edificio, o que pueda convertir un identificador de una estantería en un almacén a coordenadas que definan la ubicación de la estantería en el almacén.

En otros casos, es posible que los datos existentes que un geocodificador convierta sean datos espaciales. Por ejemplo, es posible que un geocodificador proporcionado por un usuario convierta coordenadas X e Y a datos que se adecuen a uno de los tipos de datos de DB2 Spatial Extender.

En DB2 Spatial Extender, geocodificar es simplemente la operación por la que un geocodificador convierte la entrada en salida, convirtiendo las direcciones en coordenadas, por ejemplo.

## Modalidades

Un geocodificador se puede ejecutar en dos modalidades:

- En modalidad de proceso por lotes, un geocodificador intenta, en una sola operación, convertir todos sus datos de entrada desde una sola tabla. Por ejemplo, en modalidad de proceso por lotes, DB2SE\_USA\_GEOCODER intenta convertir todas las direcciones en una sola tabla (o, alternativamente, todas las direcciones en un subconjunto especificado de filas de la tabla).
- En modalidad automática, un geocodificador convierte los datos tan pronto como se ha insertado o actualizado en una tabla. El geocodificador se activa mediante los desencadenantes de inserción y actualización que están definidos en la tabla.

## Proceso de geocodificación

La geocodificación es una de las diversas operaciones por las que el contenido de una columna espacial de una tabla de DB2 se obtiene a partir de otros datos. Esta explicación hace referencia a estas operaciones en conjunto como un proceso de geocodificación. Los procesos de geocodificación pueden variar de un geocodificador a otro. Por ejemplo, DB2SE\_USA\_GEOCODER busca archivos de direcciones conocidas para determinar si cada dirección que recibe como dato de entrada coincide con una dirección conocida en un grado determinado. Debido a que las direcciones conocidas son como material de referencia que la gente buscará cuando realicen la búsqueda, estas direcciones se denominan colectivamente datos de referencia. Es posible que otros geocodificadores no necesiten datos de referencia; es posible que verifiquen sus datos de entrada de otras maneras. El proceso de geocodificación en el que DB2SE\_USA\_GEOCODER participa se realiza de la manera siguiente:

1. DB2SE\_USA\_GEOCODER realiza operaciones que se han diseñado para llevar a cabo:
  - a. DB2SE\_USA\_GEOCODER analiza cada dirección que recibe como datos de entrada.
  - b. DB2SE\_USA\_GEOCODER busca los datos de referencia para nombres de calle que, en un grado determinado, se parecen al nombre de calle en la dirección analizada. Restringe su búsqueda a calles en el área designada por el código postal de la dirección.
  - c. Si la búsqueda es satisfactoria, DB2SE\_USA\_GEOCODER determina si alguna dirección de las calles que ha encontrado coincide en un grado determinado con la dirección analizada.
  - d. Si DB2SE\_USA\_GEOCODER encuentra una coincidencia, geocodifica la dirección analizada. De lo contrario, devuelve un valor nulo.
2. Si DB2SE\_USA\_GEOCODER geocodifica la dirección analizada, DB2 coloca las coordenadas resultantes en una columna espacial designada.
3. Si DB2SE\_USA\_GEOCODER está geocodificando en modalidad de proceso por lotes, DB2 Spatial Extender emite una confirmación (a) cada vez que

DB2SE\_USA\_GEOCODER acaba de procesar cierto número de registros de entrada o (b) después de que DB2SE\_USA\_GEOCODER acabe de procesar todos sus datos de entrada.

## Tareas del usuario

En DB2 Spatial Extender, las tareas que pertenecen a la geocodificación son:

- Prescripción de cómo algunas partes del proceso de geocodificación se deben ejecutar para una columna espacial determinada; por ejemplo, estableciendo el grado mínimo en el que deben coincidir los nombres de calle en los registros de entrada y los nombres de calle en los datos de referencia, estableciendo el grado mínimo en el que las direcciones en los registros de entrada y las direcciones en los datos de referencia deben coincidir, y determinando cuántos registros se deben procesar antes de cada confirmación. Se puede hacer referencia a esta tarea como definición de la geocodificación o definición de las operaciones de geocodificación.
- Especificación de que los datos se deben geocodificar automáticamente cada vez que se añaden a la tabla o se actualizan en la misma. Cuando se produce la geocodificación automática, tendrán lugar las instrucciones que el usuario ha especificado cuando configuró las operaciones de geocodificación (excepto para las instrucciones que impliquen confirmaciones; éstas sólo se aplican a la geocodificación de proceso por lotes). Esta tarea se denomina configuración de un geocodificador para que se ejecute automáticamente.
- Ejecución de un geocodificador en modalidad de proceso por lotes. Si el usuario ya ha establecido las operaciones de geocodificación, sus instrucciones seguirán siendo efectivas durante cada sesión de proceso por lotes, a menos que el usuario las altere temporalmente. Si el usuario no ha establecido las operaciones de geocodificación antes de una sesión determinada, el usuario puede especificar que éstas sean efectivas para esta sesión en concreto. Se puede hacer referencia a esta tarea como ejecución de un geocodificador en modalidad de proceso por lotes y ejecución de la geocodificación en modalidad de proceso por lotes.

## Definición de las operaciones de geocodificación

DB2 Spatial Extender le permite definir, de antemano, el trabajo que es necesario realizar cuando se invoca un geocodificador.

Antes de establecer las operaciones de geocodificación para un geocodificador determinado, el ID de usuario debe tener una de las siguientes autorizaciones o privilegios:

- Autorización DATAACCESS sobre la base de datos que contiene las tablas sobre las que operará el geocodificador
- Privilegio CONTROL sobre cada tabla sobre la que opera el geocodificador
- Privilegio SELECT y privilegio UPDATE sobre cada tabla sobre la que opera el geocodificador

Puede especificar los siguientes parámetros cuando se invoca un geocodificador:

- Para qué columna el geocodificador va a proporcionar datos.
- Si los datos de entrada que el geocodificador lee de la tabla o vista deben estar limitados a un subconjunto de filas de la tabla o vista.
- El rango o número de registros que el geocodificador debe geocodificar en sesiones de proceso por lotes dentro de una unidad de trabajo.
- Requisitos para operaciones específicas del geocodificador. Por ejemplo, DB2SE\_USA\_GEOCODER puede geocodificar sólo los registros que coinciden



con los elementos correspondientes a los mismos en los datos de referencia en un grado especificado o superior. Este grado se denomina grado mínimo de coincidencia.

Debe especificar los parámetros de la lista antes de definir el geocodificador para que se ejecute en modalidad automática. De entonces en adelante, cada vez que invoque al geocodificador (no sólo automáticamente, sino también en ejecuciones de proceso por lotes), se realizarán operaciones de geocodificación según las especificaciones. Por ejemplo, si especifica que se deben geocodificar 45 registros en modalidad de proceso por lotes dentro de cada unidad de trabajo, se emitirá una confirmación después de que se geocodifique cada cuarenta y cinco registro. (Excepción: Puede alterar temporalmente las especificaciones para sesiones individuales de geocodificación de proceso por lotes.)

No tiene que establecer valores por omisión para operaciones de geocodificación antes de ejecutar el geocodificador en modalidad de proceso por lotes. En su lugar, en el momento de iniciar una sesión de proceso por lotes, puede especificar cómo se deben realizar las operaciones durante la duración de la ejecución. Si establece valores por omisión para las sesiones de proceso por lotes, puede alterarlos temporalmente, si es necesario, para sesiones individuales.

Para realizar esta tarea:

Elija el modo en que desea establecer operaciones de geocodificación:

- Invóquelo desde la ventana Configurar geocodificación del Centro de control de DB2.
- Emita el mandato `db2se setup_gc`.
- Ejecute una aplicación que llame al procedimiento almacenado `db2gse.ST_setup_geocoding`.

**Recomendaciones:** Cuando DB2SE\_USA\_GEOCODER lee un registro de un dato de dirección, intenta hacer coincidir dicho registro con un elemento correspondiente en los datos de referencia. De forma esquematizada, la forma en que este proceso tiene lugar es la siguiente: Primero, busca el dato de referencia de las calles cuyo código postal es el mismo que el código postal del registro. Si encuentra un nombre de calle que es similar al nombre en el registro en un cierto grado mínimo, o en un grado superior a este grado mínimo, continúa buscando una dirección completa. Si encuentra una dirección completa que es similar a la dirección en el registro en un cierto grado mínimo, o en un grado superior a este grado mínimo, geocodifica el registro. Si no encuentra una dirección de este tipo, devuelve un valor nulo.

El grado mínimo con el que los nombres de calle deben coincidir se denomina sensibilidad ortográfica. El grado mínimo con el que todas las direcciones deben coincidir se denomina grado mínimo de coincidencia. Por ejemplo, si la sensibilidad ortográfica es de 80, la coincidencia entre los nombres de calle debe ser exacta como mínimo en un 80 por ciento para que el geocodificador busque la dirección completa. Si el grado mínimo de coincidencia es 60, la coincidencia entre las direcciones debe ser exacta como mínimo en un 60 por ciento para que el geocodificador geocodifique el registro.

Puede especificar cuál debe ser la sensibilidad ortográfica y el grado mínimo de coincidencia en las búsquedas. Tenga en cuenta que es posible que necesite ajustar estos valores. Por ejemplo, supongamos que la sensibilidad ortográfica y el grado mínimo de coincidencia sean ambas de 95. Si las direcciones que desea

geocodificar no se han validado cuidadosamente, es altamente improbable que haya coincidencias con el 95 por ciento de exactitud. Como resultado, es probable que el geocodificador devuelva un valor nulo cuando procese estos registros. En dicho caso, es aconsejable que disminuya la sensibilidad y el grado mínimo de coincidencia, y que ejecute de nuevo el geocodificador. Los valores recomendados para el grado de sensibilidad ortográfica y el grado mínimo de coincidencia son 70 y 60, respectivamente.

Como se ha indicado al principio de esta explicación, puede determinar si los datos de entrada que lee el geocodificador de la tabla o vista se deben limitar a un subconjunto de filas de la tabla o vista. Por ejemplo, tenga en cuenta los siguientes casos:

- Invoca al geocodificador para geocodificar direcciones de una tabla en modalidad de proceso por lotes. Desafortunadamente, el grado mínimo de coincidencia es demasiado alto, lo que provoca que el geocodificador devuelva un valor nulo cuando procesa la mayoría de las direcciones. Reduzca el grado mínimo de coincidencia cuando ejecute de nuevo el geocodificador. Para limitar sus datos de entrada a las direcciones que no se han geocodificado, especifique que debe seleccionar sólo aquellas filas que contienen el valor nulo que ha devuelto anteriormente.
- El geocodificador sólo selecciona filas que se han añadido después de una fecha determinada.
- El geocodificador selecciona sólo las filas que contienen direcciones en un área determinada; por ejemplo, un conjunto de regiones o una provincia.

Como se ha indicado al inicio de esta explicación, puede determinar el número de registros que el geocodificador debe procesar en sesiones de proceso por lotes dentro de una unidad de trabajo. Puede hacer que el geocodificador procese el mismo número de registros en cada unidad de trabajo, o bien puede hacer que procese todos los registros de una tabla dentro de una única unidad de trabajo. Si elige esta última alternativa, tenga en cuenta lo siguiente:

- Tiene menos control sobre el tamaño de la unidad de trabajo de lo que le permite la alternativa anterior. En consecuencia, no puede controlar cuántos bloqueos se mantienen o cuántas entradas de anotaciones cronológicas se realizan cuando opera el geocodificador.
- Si el geocodificador encuentra un error que necesita una retrotracción, es necesario ejecutar de nuevo el geocodificador para que se ejecute para todos los registros. El consiguiente consumo de recursos puede ser alto si la tabla es muy grande y el error y la cancelación de cambios se producen una vez procesados la mayoría de los registros.

## **Configuración de un geocodificación para que se ejecute automáticamente**

Puede configurar un geocodificador para que convierta automáticamente los datos a medida que estos datos se añadan a una tabla o se actualicen en ella.

Antes de poder configurar un geocodificador para que se ejecute automáticamente:

- Debe configurar las operaciones de geocodificación para cada columna espacial que se va a llenar con los datos de salida de un geocodificador.
- El ID de usuario debe tener las siguientes formas de autorización o privilegios:
  - Autorización DBADM y DATAACCESS sobre la base de datos que contiene la tabla en la que se definirán los desencadenantes para invocar al geocodificador

- Privilegio CONTROL
- Privilegios ALTER, SELECT y UPDATE
- Los privilegios necesarios para crear desencadenantes en esta tabla.

Puede configurar un geocodificador para que se ejecute automáticamente antes de invocarlo en modalidad de proceso por lotes. Por lo tanto, es posible que la geocodificación automática preceda a la geocodificación de proceso por lotes. Si esto sucede, es probable que la geocodificación de proceso por lotes invoque al proceso de los mismos datos que se han procesado automáticamente. Esta redundancia no causará datos duplicados, porque cuando los datos espaciales se producen dos veces, el segundo grupo de datos prevalece sobre el primero. Sin embargo, esto puede degradar el rendimiento.

Antes de decidir si desea geocodificar los datos de direcciones de una tabla en modalidad de proceso por lotes o en modalidad automática, considere lo siguiente:

- El rendimiento es mejor en la geocodificación de proceso por lotes que en la geocodificación automática. Una sesión de proceso por lotes se abre con una inicialización y finaliza con una limpieza. En la geocodificación automática, cada dato se geocodifica en una única operación que empieza con la inicialización y finaliza con la limpieza.
- En conjunto, es probable que una columna espacial que se ha llenado mediante una geocodificación automática esté más actualizada que una columna espacial que se haya llenado mediante una geocodificación de proceso por lotes. Después de cada sesión de proceso por lotes, los datos de direcciones se pueden acumular y permanecer sin geocodificar hasta la siguiente sesión. Pero si ya geocodificación automática ya está habilitada, los datos de direcciones se geocodifican tan pronto como se almacenan en la base de datos.

Para realizar esta tarea:

Elija qué método desea utilizar para configurar la geocodificación automática:

- Hágalo desde la ventana Configurar geocodificación o desde la ventana Geocodificación del Centro de control de DB2.
- Emita el mandato `db2se enable_autogc`.
- Ejecute una aplicación que llame al procedimiento almacenado `db2gse.ST_enable_autogeocoding`.

## Ejecución de un geocodificador en modalidad de proceso por lotes

Cuando se ejecuta un geocodificador en modalidad de proceso por lotes, debe convertir varios registros en datos espaciales que van en una columna específica.

Antes de ejecutar un geocodificador en modalidad de proceso por lotes, el ID de usuario debe tener una de las siguientes formas de autorización o privilegios:

- Autorización DATAACCESS sobre la base de datos que contiene la tabla cuyos datos se van a geocodificar
- Privilegio CONTROL sobre esta tabla
- Privilegio UPDATE sobre esta tabla

También necesita el privilegio SELECT sobre esta tabla, de forma que pueda especificar el número de registros que se procesarán antes de cada confirmación. Si especifica cláusulas WHERE para limitar las filas en las que operará el

geocodificador, es posible que también necesite el privilegio SELECT sobre cualquier tabla y vista a la que haga referencia en dichas cláusulas. Consulte al administrador de la base de datos.

En cualquier momento antes de ejecutar un geocodificador para que llene una columna espacial determinada, puede configurar operaciones de geocodificación para dicha columna. La definición de las operaciones implica especificar cómo se van a cumplir ciertos requisitos cuando se ejecute el geocodificador. Por ejemplo, suponga que necesita que DB2 Spatial Extender emita una confirmación cada vez que el geocodificador procese 100 registros de entrada. Cuando haya configurado las operaciones, especificará 100 como el número necesario.

Cuando ya esté preparado para ejecutar el geocodificador, puede alterar temporalmente cualquiera de los valores que ha especificado cuando configuró las operaciones. Los valores con los que ha alterado temporalmente estos valores sólo serán efectivos durante la duración de la ejecución.

Si no ha configurado las operaciones, debe, cada vez que esté preparado para ejecutar el geocodificador, especificar cómo se cumplirán los requisitos durante la ejecución.

Para realizar esta tarea... :

Elija cómo invocar un geocodificador para ejecutar en modalidad de proceso por lotes:

- Invóquelo desde la ventana Ejecutar geocodificación del Centro de control de DB2.
- Emita el mandato `db2se run_gc`.
- Ejecute una aplicación que llame al procedimiento almacenado `db2gse.ST_run_geocoding`.

---

## Capítulo 11. Utilización de índices y vistas para acceder a datos espaciales

Antes de consultar columnas espaciales, puede crear índices y vistas que facilitarán el acceso a ellas. Este capítulo:

- Describe la naturaleza de los índices que Spatial Extender utiliza para activar el acceso a los datos espaciales
- Explica cómo crear estos índices
- Explica cómo utilizar las vistas para acceder a los datos espaciales

---

### Tipos de índices espaciales

Para obtener un buen rendimiento, es importante tener índices eficientes definidos para las columnas de las tablas base en una base de datos. El rendimiento de la consulta está relacionado directamente con la velocidad con que se pueden encontrar los valores durante la consulta. Las consultas que utilizan un índice se pueden ejecutar más rápidamente y proporcionan mejoras de rendimiento significativas.

Habitualmente, en las consultas espaciales intervienen dos o más dimensiones. Por ejemplo, en una consulta espacial, tal vez desee saber si un punto está incluido dentro de un área (polígono). Debido a la naturaleza multidimensional de las consultas espaciales, la indexación por árbol binario nativo de DB2® no es eficaz para estas consultas.

Las consultas espaciales pueden utilizar los siguientes tipos de índice:

- Índices reticulares espaciales

La tecnología de indexación de DB2 Spatial Extender utiliza la *indexación reticular*, que está diseñada para indexar datos espaciales multidimensionales y columnas espaciales. DB2 Spatial Extender proporciona un índice reticular que está optimizado para datos de dos dimensiones en una proyección plana de la Tierra.

- Índices Voronoi geodésicos

DB2 Geodetic Data Management Feature proporciona soporte para un nuevo método de acceso espacial que permite crear índices en columnas que contengan datos geodésicos multidimensionales. Un índice Voronoi geodésico es más apropiado que un índice reticular para los datos geodésicos porque considera que la tierra es una esfera continua sin distorsiones alrededor de los polos o bordes en el meridiano número 180.

---

### Índices reticulares espaciales

Los índices mejoran el rendimiento de las consultas de la aplicación, especialmente cuando la tabla o las tablas consultas contienen muchas filas. Si crea índices apropiados que el optimizador de consultas pueda elegir para ejecutar la consulta, reducirá de forma significativa el número de filas que se procesarán.

DB2 Spatial Extender proporciona un índice reticular optimizado para dos datos dimensionales. El índice se crea en las dimensiones X e Y de una geometría.

Los siguientes aspectos de un índice reticular le ayudarán a comprender:

- La generación del índice
- El uso de funciones espaciales en una consulta
- Cómo una consulta utiliza un índice reticular espacial

## Generación de índices reticulares espaciales

Spatial Extender genera un índice reticular espacial utilizando el rectángulo delimitador mínimo (MBR) de una geometría.

Para muchas geometrías, el MBR es un rectángulo que rodea la geometría.

Un índice reticular espacial divide una región en retículas cuadradas lógicas con un tamaño fijo especificado por el usuario al crear el índice. El índice espacial se construye en una columna espacial creando una o más entradas para las intersecciones del MBR de cada geometría con las celdas reticulares. Una entrada de índice consta del identificador de celda reticular, del MBR de la geometría del identificador interno de la fila que contiene la geometría.

Puede definir hasta tres niveles de índices espaciales (niveles reticulares). La utilización de varios niveles reticulares es ventajosa porque le permite optimizar el índice para diferentes tamaños de datos espaciales.

Si una geometría forma intersección con cuatro o más celdas reticulares, la geometría se promociona al siguiente nivel más grande. En general, las geometrías más grandes se indexarán en los niveles más grandes. Si una geometría forma intersección con 10 o más celdas reticulares en el tamaño de retícula más grande, se utiliza un nivel de índice de desbordamiento definido por el sistema. Este nivel de desbordamiento impide la generación de demasiadas entradas de índice. Para obtener el mejor rendimiento, defina los tamaños de retícula para evitar el uso de este nivel de desbordamiento.

Por ejemplo, si hay varios niveles reticulares, el algoritmo de creación de índices intenta utilizar el nivel reticular más bajo posible para proporcionar la resolución más fina para los datos indexados. Cuando una geometría forma intersección con más de cuatro celdas reticulares en un nivel determinado, se promociona al nivel superior siguiente, (siempre que exista otro nivel). Por lo tanto, un índice espacial que tenga tres niveles reticulares de 10.0, 100.0 y 1000.0 formará primero intersección con cada retícula de nivel 10.0. Si una geometría forma intersección con más de cuatro celdas reticulares con un tamaño de 10.0, se promociona y forma intersección con la retícula de nivel 100.0. Si más de cuatro intersecciones se producen en el nivel 100.0, la geometría se promociona al nivel 1000.0. Si más de 10 intersecciones dan como resultado en el nivel 1000.0, la geometría se indexa en el nivel de desbordamiento.

## Uso de funciones espaciales en una consulta

El optimizador de DB2 puede utilizar un índice reticular espacial cuando una consulta contiene una de las funciones siguientes en su cláusula WHERE:

- ST\_Contains
- ST\_Crosses
- ST\_Distance
- ST\_EnvIntersects
- EnvelopesIntersect
- ST\_Equals

- ST\_Intersects
- ST\_MBRIntersects
- ST\_Overlaps
- ST\_Touches
- ST\_Within

## Cómo una consulta utiliza un índice reticular espacial

Cuando el optimizador de consultas elige un índice reticular espacial, la ejecución de la consulta utiliza un proceso de filtrado que consta de varios pasos.

El proceso de filtro incluye los pasos siguientes:

1. Determinar las celdas reticulares que forman intersección con la ventana de consulta. La *ventana de consulta* es la geometría que le interesa y que especifica como segundo parámetro en una función espacial (consulte los ejemplos más adelante).
2. Explorar el índice para localizar entradas que tengan identificadores de celdas reticulares coincidentes.
3. Comparar los valores de MBR de la geometría en las entradas de índice con la ventana de consulta y descartar los valores que estén fuera de la ventana de consulta.
4. Realizar análisis adicionales si es preciso. Podrían realizarse análisis adicionales en el conjunto de geometrías candidato de los pasos anteriores para determinar si reúnen los requisitos de la función espacial (ST\_Contains, ST\_Distance, etcétera). La función espacial EnvelopesIntersect omite este paso y normalmente ofrece el mejor rendimiento.

Los siguientes ejemplos de consultas espaciales tienen un índice reticular espacial en la columna C.GEOMETRY:

```
SELECT nombre
FROM condados AS c
WHERE EnvelopesIntersect(c.geometry, -73.0, 42.0, -72.0, 43.0, 1) = 1
```

```
SELECT nombre
FROM condados AS c
WHERE ST_Intersects(c.geometry, :geometry2) = 1
```

En el primer ejemplo, los cuatro valores de coordenadas definen la ventana de consulta. Estos valores de coordenadas especifican las esquinas inferior izquierda y superior derecha (42.0 -73.0 y 43.0 -72.0) de un rectángulo.

En el segundo ejemplo, Spatial Extender calcula el MBR de la geometría especificada por la variable del sistema principal :geometry2 y la utiliza como ventana de consulta.

Al crear un índice reticular espacial, deberá especificar los tamaños de retícula apropiados para los tamaños de ventana de consulta más habituales que la aplicación espacial suele utilizar. Si un tamaño de retícula es superior, deben explorarse las entradas de índice de las geometrías que estén fuera de la ventana de consulta, porque se encuentran en celdas reticulares que forman intersección con la ventana de consulta y dichas exploraciones adicionales disminuyen el rendimiento. Sin embargo, un tamaño reticular inferior podría generar más entradas de índice para cada geometría, por lo que deberán explorarse más entradas de índice y el rendimiento de la consulta también se verá afectado.

DB2 Spatial Extender proporciona el programa de utilidad Index Advisor, que analiza los datos de las columnas espaciales y recomienda tamaños de retícula apropiados para los tamaños de ventana de consulta habituales.

---

## Consideraciones sobre el número de niveles de índice y tamaños de retícula

Utilice Index Advisor para determinar los tamaños de retícula apropiados para los índices reticulares espaciales, porque es la mejor manera de ajustar los índices y mejorar la eficacia de las consultas espaciales.

### Número de niveles de índice

Puede utilizar hasta tres niveles reticulares.

Para cada nivel reticular de un índice reticular espacial, se realiza una búsqueda de índice por separado durante una consulta espacial. Por lo tanto, si tiene más niveles reticulares, la consulta será menos eficaz.

Si los valores contenidos en la columna espacial tienen aproximadamente el mismo tamaño relativo, utilice un solo nivel reticular. Sin embargo, una columna espacial típica no contiene geometrías con el mismo tamaño relativo, aunque las geometrías de una columna espacial pueden agruparse de acuerdo con el tamaño. El usuario debe asociar los tamaños de retícula con estas agrupaciones de geometrías.

Por ejemplo, suponga que tiene una tabla con parcelas de terrenos en una columna espacial que contiene agrupaciones de pequeñas parcelas urbanas rodeadas por parcelas rurales mayores. Debido a que los tamaños de las parcelas se pueden agrupar en dos grupos (pequeñas parcelas urbanas y parcelas rurales mayores), especificaría dos niveles reticulares para el índice reticular espacial.

### Tamaños de las celdas reticulares

La norma general es disminuir lo máximo posible los tamaños de retícula para obtener la resolución más alta y al mismo tiempo minimizar el número de entradas de índice.

- Se debe utilizar un valor pequeño para el tamaño de retícula más fino con el propósito de optimizar el índice general para las geometrías pequeñas de la columna. Esto evita el coste de evaluar geometrías que no están en el área de búsqueda. Sin embargo, el tamaño de retícula más fino también produce el mayor número de entradas de índice. En consecuencia, el número de entradas de índice procesadas durante el tiempo de la consulta aumenta, así como la cantidad de almacenamiento necesario para el índice. Estos factores reducen el rendimiento general.
- Mediante la utilización de tamaños de retícula más grandes, se puede optimizar más el índice para geometrías más grandes. Los tamaños de retícula más grandes producen menos entradas de índice para geometrías grandes que lo haría el tamaño de retícula más fino. En consecuencia, se reducen los requisitos de almacenamiento para el índice, aumentando el rendimiento general.

Las siguientes figuras muestran los efectos de los diferentes tamaños de retícula.

La Figura 13 en la página 79 muestra un mapa de parcelas de terreno, en el que cada parcela se representa por medio de una geometría de polígonos. El rectángulo negro representa una ventana de consulta. Supongamos que desea encontrar todas



las geometrías cuyo MBR forma intersección con la ventana de consulta. La Figura 13 muestra que 28 geometrías (resaltadas en rosa) tienen un MBR que forma intersección con la ventana de consulta.

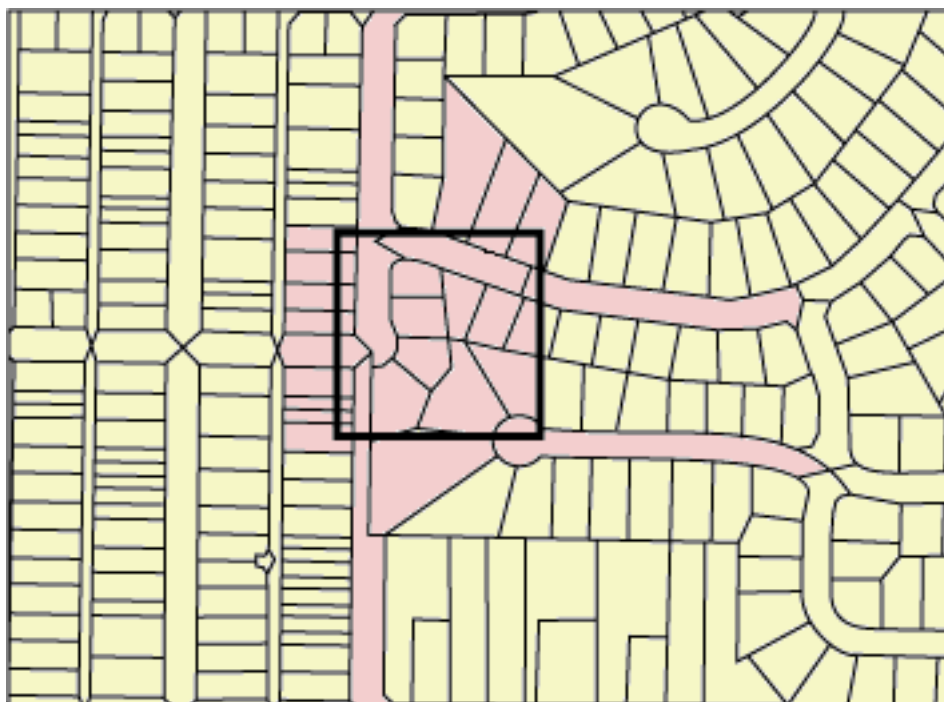


Figura 13. Parcelas de terreno en un vecindario

La Figura 14 en la página 80 muestra un tamaño de retícula pequeño (25) que se adapta mejor a la ventana de consulta.

- La consulta devuelve sólo las 28 geometrías que están resaltadas, pero debe examinar y descartar tres geometrías adicionales cuyos MBR forman intersección con la ventana de consulta.
- Este tamaño de retícula pequeño produce muchas entradas por geometría. Durante su ejecución, la consulta accede a todas las entradas de índice para estas 31 geometrías. La Figura 14 en la página 80 muestra 256 celdas reticulares que cubren la ventana de consulta. Sin embargo, la ejecución de la consulta accede a 578 entradas de índice porque muchas geometrías están indexadas con las mismas celdas reticulares.

Para esta ventana de consulta, este tamaño de retícula pequeño produce un número excesivo de entradas de índice para explorar.

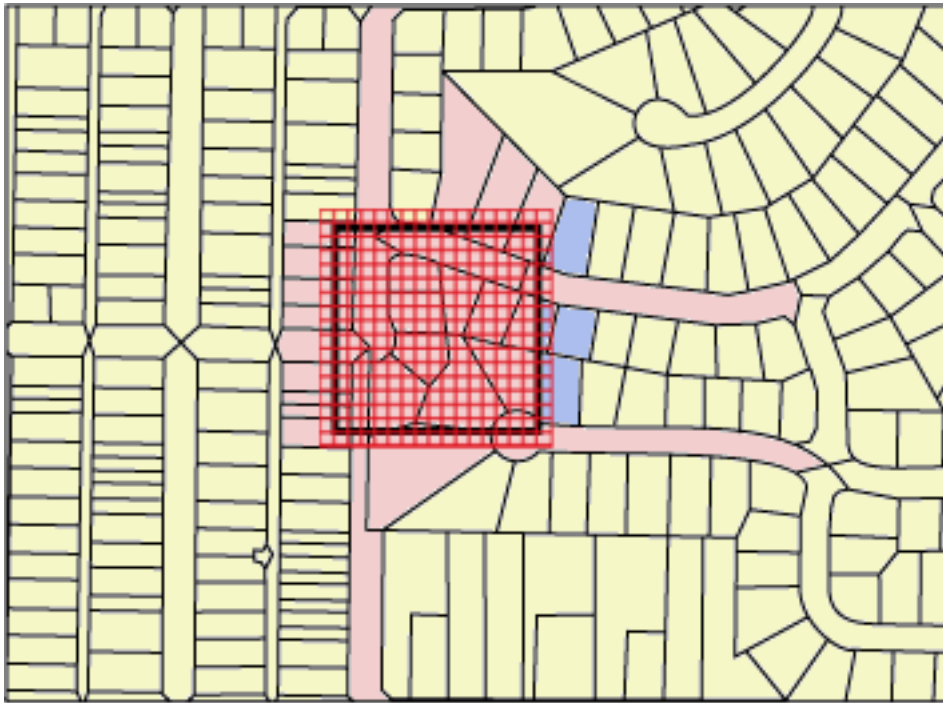


Figura 14. Tamaño de retícula pequeño (25) en parcelas de terreno

Figura 15 en la página 81 muestra un tamaño de retícula grande (400) que abarca un área considerablemente mayor con muchas más geometrías que la ventana de consulta.

- Este tamaño de retícula grande produce sólo una entrada de índice por geometría, pero la consulta debe examinar y descartar 59 geometrías adicionales cuyos MBR forman intersección con la celda reticular.
- Durante su ejecución, la consulta accede a todas las entradas de índice para las 28 geometrías que forman intersección con la ventana de consulta, además de las entradas de índice para las 59 geometrías adicionales, para un total de 112 entradas de índice.

Para esta ventana de consulta, este tamaño de retícula grande produce un número excesivo de geometrías para examinar.

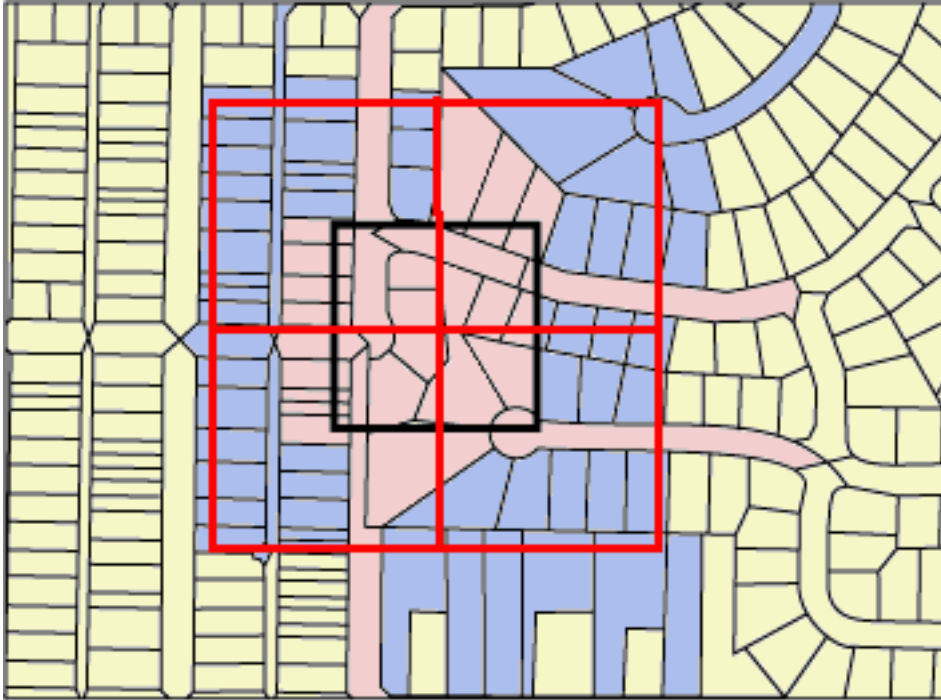


Figura 15. Tamaño de retícula grande (400) en parcelas de terreno

La Figura 16 en la página 82 muestra un tamaño de retícula medio (100) que se adapta mejor a la ventana de consulta.

- La consulta devuelve sólo las 28 geometrías que están resaltadas, pero debe examinar y descartar cinco geometrías adicionales cuyos MBR forman intersección con la ventana de consulta.
- Durante su ejecución, la consulta accede a todas las entradas de índice para las 28 geometrías que forman intersección con la ventana de consulta, además de las entradas de índice para las 5 geometrías adicionales, para un total de 91 entradas de índice.

Para esta ventana de consulta, este tamaño de retícula medio es el mejor porque produce un número significativamente inferior de entradas de índice que el tamaño de retícula pequeño y la consulta examina menos geometrías adicionales que el tamaño de retícula grande.

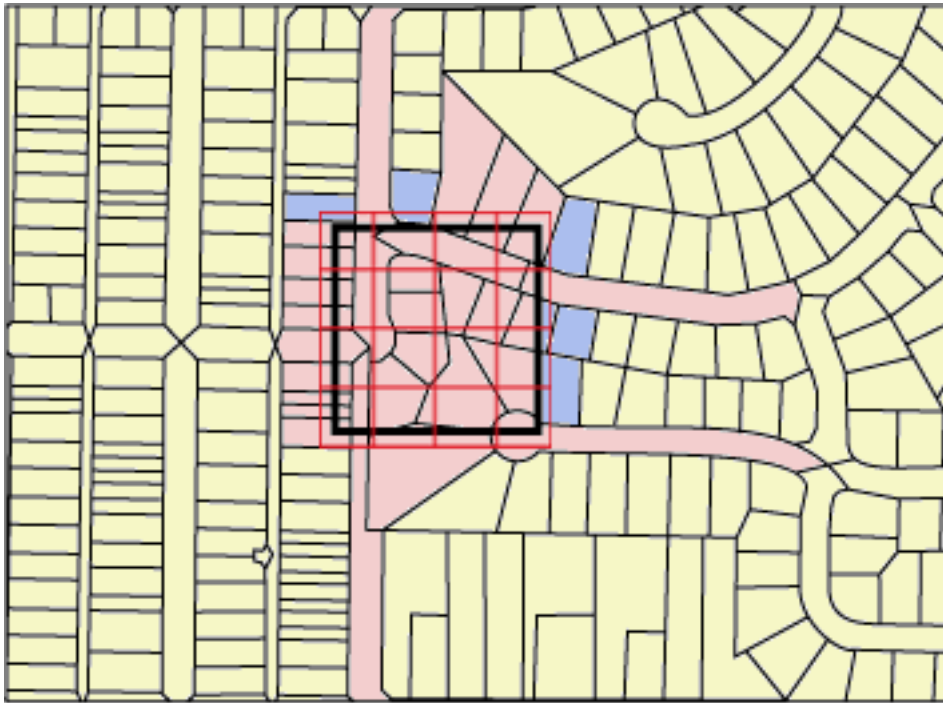


Figura 16. Tamaño de retícula medio (100) en parcelas de terreno

## Creación de índices reticulares espaciales

Cree índices reticulares espaciales para definir índices reticulares bidimensionales en columnas espaciales con el fin de ayudar a optimizar las consultas espaciales.

Para crear un índice reticular espacial:

- Su ID de usuario debe tener las autorizaciones necesarias para la sentencia CREATE INDEX de SQL de DB2. El ID de usuario debe tener como mínimo uno de los siguientes privilegios o autorizaciones:
  - DBADM sobre la base de datos donde reside la tabla que contiene la columna
  - Los dos privilegios o autorizaciones siguientes:
    - Uno de los siguientes privilegios:
      - Privilegio CONTROL sobre la tabla
      - Privilegio INDEX sobre la tabla
    - Uno de los siguientes privilegios o autorizaciones en el esquema:
      - Autorización IMPLICIT\_SCHEMA sobre la base de datos, si el esquema del índice no existe
      - Privilegio CREATEIN sobre el esquema, si el nombre de esquema del índice hace referencia a un esquema existente
- Debe conocer los valores que desea especificar para el nombre calificado al completo del índice reticular espacial y los tres tamaños de retícula que utilizará el índice.

**Recomendaciones:**

- Para crear un índice reticular espacial en una columna, utilice Index Advisor para determinar los parámetros del índice. Index Advisor puede analizar datos de columnas espaciales y sugerir tamaños de retícula apropiados para el índice reticular espacial utilizado.
- Si prevé realizar una carga inicial de datos en la columna, deberá crear el índice reticular espacial después de finalizar el proceso de carga. De esta forma, puede seleccionar los tamaños óptimos de las celdas reticulares de acuerdo con las características de los datos, utilizando Index Advisor. Además, el rendimiento del proceso de carga será mayor si carga los datos antes de crear el índice, pues no será necesario realizar el mantenimiento del índice reticular espacial durante el proceso de carga.

**Restricción:**

Existen las mismas restricciones para crear índices mediante la sentencia CREATE INDEX cuando crea un índice reticular espacial. Es decir, la columna en la que se crea un índice debe ser una columna de tabla base no una columna de vista ni una columna deseudónimo. El sistema de base de datos de DB2 resolverá los alias en el proceso.

Los índices reticulares espaciales se crean para mejorar el rendimiento de las consultas en las columnas espaciales.

Cuando se crea un índice reticular espaciales se debe proporcionar la siguiente información:

- Un nombre
- El nombre de la columna espacial en la que se va a definir
- La combinación de los tres tamaños de retícula ayuda a optimizar el rendimiento reduciendo el número total de entradas de índice y el número de las mismas que deben explorarse para satisfacer una consulta.

Puede crear un índice reticular espacial de cualquiera de las maneras siguientes:

- Utilice la ventana Spatial Extender del Centro de control de DB2.
- Utilice la sentencia CREATE INDEX de SQL con la extensión db2gse.spatial\_index en la cláusula EXTEND USING.
- Utilice una herramienta GIS que funciona con DB2 Spatial Extender. Si utiliza dicha herramienta para crear el índice, la herramienta emitirá la sentencia CREATE INDEX de SQL apropiada.

Este tema presenta los pasos para los dos primeros métodos. Para obtener información sobre la utilización de una herramienta GIS para crear un índice reticular espacial, consulte la documentación proporcionada con dicha herramienta.

Para realizar esta tarea... :

## Creación de un índice espacial utilizando SQL CREATE INDEX

1. Determine la sentencia CREATE INDEX utilizando la cláusula EXTEND USING y la extensión de índice reticular db2gse.spatial\_index. Por ejemplo, la siguiente sentencia crea el índice reticular espacial TERRIDX para la tabla BRANCHES que tiene una columna espacial TERRITORY.

```
CREATE INDEX terridx
  ON branches (territory)
  EXTEND USING db2gse.spatial_index (1.0, 10.0, 100.0)
```

2. Emita el mandato CREATE INDEX en el Editor de mandatos de DB2, la ventana Mandatos de DB2 o el procesador de línea de mandatos de DB2.

---

## Sentencia CREATE INDEX para un índice reticular espacial

Utilice la sentencia CREATE INDEX con la cláusula EXTEND USING para crear un índice reticular espacial.

### Sintaxis

```
▶ CREATE INDEX esquema_índice. nombre_índice ON  
▶ esquema_tabla. nombre_tabla (nombre_columna) EXTEND USING  
▶ db2gse.spatial_index (tamaño_retícula_fino, tamaño_retícula_medio  
▶ , tamaño_retícula_medio)
```

### Parámetros

#### esquema\_índice.

Nombre del esquema al que debe pertenecer el índice que está creando. Si no especifica un nombre, el sistema de base de datos de DB2 utiliza el nombre de esquema que se ha almacenado en el registro especial CURRENT SCHEMA.

#### nombre\_índice

Nombre, no calificado, del índice reticular que está creando.

#### esquema\_tabla.

Nombre del esquema al que pertenece la tabla donde reside *nombre\_columna*. Si no especifica un nombre, DB2 utiliza el nombre de esquema que está almacenado en el registro especial CURRENT SCHEMA.

#### nombre\_tabla

Nombre, no calificado, de la tabla donde reside *nombre\_columna*.

#### nombre\_columna

Nombre de la columna espacial para la que se crea el índice reticular espacial.

#### tamaño\_retícula\_fino, tamaño\_retícula\_medio, tamaño\_retícula\_grueso

Tamaños de retícula para el índice reticular espacial. Estos parámetros deben cumplir estas condiciones:

- *tamaño\_retícula\_fino* debe ser mayor que 0.
- *tamaño\_retícula\_medio* debe ser mayor que *tamaño\_retícula\_fino* o igual a 0.
- *tamaño\_retícula\_grueso* debe ser mayor que *tamaño\_retícula\_medio* o igual a 0.

Cuando crea el índice reticular espacial utilizando el Centro de control o la sentencia CREATE INDEX, la validez de los tamaños de retícula se comprueba al indexar la primera geometría. Por lo tanto, si los tamaños de retícula que especifica no cumplen las condiciones de sus valores, se emite una condición de error tal como se describe en estas situaciones:

- Si todas las geometrías de la columna espacial son nulas, Spatial Extender crea satisfactoriamente el índice sin verificar la validez de los tamaños de retícula. Spatial Extender valida los tamaños de retícula cuando el usuario inserta o

actualiza una geometría que no sea nula en esa columna espacial. Si los tamaños de retícula especificados no son válidos, se produce un error al insertar o actualizar la geometría no nula.

- Si hay geometrías no nulas en la columna espacial cuando el usuario crea el índice, Spatial Extender valida los tamaños de retícula en ese momento. Si los tamaños de retícula especificados no son válidos, se produce un error inmediatamente y no se crea el índice reticular espacial.

## Ejemplo

En el siguiente ejemplo, la sentencia CREATE INDEX crea el índice reticular espacial TERRIDX en la columna espacial TERRITORY de la tabla BRANCHES:

```
CREATE INDEX terridx
  ON branches (territory)
  EXTEND USING db2gse.spatial_index (1.0, 10.0, 100.0)
```

---

## Ajuste de índices reticulares espaciales con Index Advisor

### Ajuste de índices reticulares espaciales con Index Advisor—Visión general

DB2® Spatial Extender proporciona un programa de utilidad denominado Index Advisor, que puede utilizar para:

- Determinar los tamaños de retícula apropiados para sus índices reticulares espaciales.

Index Advisor analiza las geometrías en una columna espacial y recomienda tamaños de retícula óptimos para el índice reticular espacial.

- Analice un índice reticular existente.

Index Advisor puede recopilar y visualizar estadísticas a partir de las que puede determinar cómo facilitarán la recuperación de los datos espaciales los tamaños de celda reticular actuales.

### Determinación de los tamaños de retícula para un índice reticular espacial

Requisitos previos

Para poder analizar los datos que desea indexar:

- Su ID de usuario debe tener privilegio SELECT sobre esta tabla.
- Si la tabla tiene más de un millón de filas, tal vez desee utilizar la cláusula ANALYZE para analizar un subconjunto de las filas para disponer de suficiente tiempo de proceso. Deberá tener disponible un espacio de tablas USER TEMPORARY para utilizar la cláusula ANALYZE. Establezca el tamaño de página de este espacio de tablas en al menos 8 KB y asegúrese de que dispone de privilegios USE sobre él. Por ejemplo, las siguientes sentencias DDL crean una agrupación de almacenamientos intermedios con el mismo tamaño de página que el espacio de tablas temporal y otorgan el privilegio USE a cualquiera:

```
CREATE BUFFERPOOL bp8k SIZE 1000 PAGESIZE 8 K;
CREATE USER TEMPORARY TABLESPACE usertempts
  PAGESIZE 8K
```

```

MANAGED BY SYSTEM USING ('c:\tempts')
BUFFERPOOL bp8k
GRANT USE OF TABLESPACE usertempts TO PUBLIC;

```

Otra alternativa consiste en utilizar el Centro de control de DB2 para crear un espacio en la tabla del usuario y la agrupación de almacenamientos intermedios correspondiente.

Para crear un índice reticular espacial en una columna, utilice Index Advisor para determinar los tamaños de retícula apropiados.

Para realizar esta tarea... :

Determine los tamaños de retícula apropiados para un índice reticular espacial:

1. Utilice Index Advisor para determinar un tamaño de celda reticular recomendado para el índice que desea crear.
  - a. Escriba el mandato que invoca Index Advisor con la palabra clave ADVISE para solicitar los tamaños de las celdas de la retícula. Por ejemplo, para invocar a Index Advisor para la columna SHAPE en la tabla COUNTIES, escriba lo siguiente:

```

gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY
STATISTICS FOR COLUMN ID_usuario.counties(shape) ADVISE

```

**Restricción:** Si entra el mandato gseidx anterior desde un indicador del sistema operativo, deberá escribir todo el mandato en una sola línea. De forma alternativa, puede ejecutar los mandatos gseidx desde un archivo CLP, lo que permite dividir el mandato en varias líneas.

Index Advisor devolverá los tamaños de celda reticular recomendados. Por ejemplo, el mandato gseidx anterior con la palabra clave ADVISE devuelve los siguientes tamaños de celda recomendados para la columna SHAPE:

Tamaño de la ventana de consulta	Tamaños de retícula recomendados			Coste
-----	-----	-----	-----	-----
0.1	0.7,	2.8,	14.0	2.7
0.2	0.7,	2.8,	14.0	2.9
0.5	1.4,	3.5,	14.0	3.5
1	1.4,	3.5,	14.0	4.8
2	1.4,	3.5,	14.0	8.2
5	1.4,	3.5,	14.0	24
10	2.8,	8.4,	21.0	66
20	4.2,	14.7,	37.0	190
50	7.0,	14.0,	70.0	900
100	42.0,	0,	0	2800

- b. Elija un tamaño apropiado para la ventana de consulta en la salida de gseidx, basado en la anchura de las coordenadas que visualiza en la pantalla.  
 En este ejemplo, los valores de latitud y longitud en grados decimales representan las coordenadas. Si la pantalla de mapa típica tiene una anchura de unos 0,5 grados (aproximadamente 55 kilómetros), vaya a la fila que tiene el valor de 0,5 en la columna Tamaño de la ventana de consulta. Esta fila tiene unos tamaños de retícula recomendados de 1,4, 3,5 y 14,0.
2. Cree el índice con los tamaños de retícula recomendados. Para el ejemplo del paso anterior, puede ejecutar la siguiente sentencia de SQL:

```

CREATE INDEX counties_shape_idx ON userID.counties(shape)
EXTEND USING DB2GSE.SPATIAL_INDEX(1.4,3.5,14.0);

```



## Análisis de estadísticas de índices reticulares espaciales

Requisitos previos

Para poder analizar los datos que desea indexar:

- Su ID de usuario debe tener privilegio SELECT sobre esta tabla.
- Si la tabla tiene más de un millón de filas, tal vez desee utilizar la cláusula ANALYZE para analizar un subconjunto de las filas para disponer de suficiente tiempo de proceso. Deberá tener disponible un espacio de tablas USER TEMPORARY para utilizar la cláusula ANALYZE. Establezca el tamaño de página de este espacio de tablas en al menos 8 KB y asegúrese de que dispone de privilegios USE sobre él. Por ejemplo, las siguientes sentencias de DDL crean una agrupación de almacenamientos intermedios con el mismo tamaño de página que el espacio de tablas temporal del usuario y otorgan el privilegio USE a cualquiera:

```
CREATE BUFFERPOOL bp8k SIZE 1000 PAGESIZE 8 K;  
CREATE USER TEMPORARY TABLESPACE usertempts  
    PAGESIZE 8K  
    MANAGED BY SYSTEM USING ('c:\tempts')  
    BUFFERPOOL bp8k  
GRANT USE OF TABLESPACE usertempts TO PUBLIC;
```

Otra alternativa consiste en utilizar el Centro de control de DB2 para crear un espacio en la tabla del usuario y la agrupación de almacenamientos intermedios correspondiente.

Las estadísticas sobre un índice reticular espacial existente pueden indicarle si el índice es eficaz o si debe sustituirse por un índice más eficaz. Utilice Index Advisor para obtener estas estadísticas y, si es necesario, para sustituir el índice.

**Consejo:** Tan importante como ajustar el índice lo es comprobar si se utiliza para las consultas. Para determinar si se está utilizando un índice espacial, ejecute Visual Explain en el Centro de control de DB2 o una herramienta de línea de mandatos como db2exfmt para la consulta. En la sección “Plan de acceso” de la salida de explicación, si ve un operador de EISCAN y el nombre del índice espacial significa que la consulta utiliza el índice.

Para realizar esta tarea:

Obtenga estadísticas para un índice reticular y, si es necesario, sustituya el índice:

1. Haga que Index Advisor recopile estadísticas basándose en los tamaños de celda reticular del índice existente. Puede solicitar estadísticas para todos los datos o para un subconjunto de todos los datos.
  - Para obtener estadísticas para los datos indexados en un subconjunto de filas, entre el mandato gseidx y especifique la palabra clave ANALYZE y sus parámetros, además de la cláusula existing-index y de la palabra clave DETAIL. Puede especificar el número o el porcentaje de filas que Index Advisor debe analizar para obtener estadísticas. Por ejemplo, para obtener estadísticas para un subconjunto de los datos indexados por el índice COUNTIES\_SHAPE\_IDX, escriba:

```
gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY  
STATISTICS FOR INDEX id_usuario.counties_shape_idx DETAIL ANALYZE 25 PERCENT  
ADVISE
```

- Para obtener estadísticas para todos los datos indexados, entre el mandato gseidx y especifique la cláusula existing-index. Incluya la palabra clave DETAIL. Por ejemplo, para invocar a Index Advisor para COUNTIES\_SHAPE\_IDX, escriba:

```
gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY
STATISTICS FOR INDEX ID_usuario.counties_shape_idx DETAIL
SHOW HISTOGRAM ADVISE
```

Index Advisor devuelve estadísticas, un histograma de los datos y tamaños de celda recomendados para el índice existente. Por ejemplo, el mandato gseidx anterior para todos los datos indexados por COUNTIES\_SHAPE\_IDX devuelve las estadísticas siguientes:

Nivel reticular 1

-----

```
Tamaño de retícula                : 0.5
Número de geometrías              : 2936
Número de entradas de índice      : 12197
```

```
Número de celdas reticulares ocupadas      : 2922
Proporción de entradas de índice/geometría : 4.154292
Proporción de geometrías/celda reticular    : 1.004791
Número máximo de geometrías por celda reticular : 14
Número mínimo de geometrías por celda reticular : 1
```

Entradas de índice:	1	2	3	4	10
Absoluto :	86	564	72	1519	695
Porcentaje (%):	2.93	19.21	2.45	51.74	23.67

Nivel reticular 2

-----

```
Tamaño reticular                : 0.0
No hay geometrías indexadas en este nivel.
```

Nivel reticular 3

-----

```
Tamaño reticular                : 0.0
No hay geometrías indexadas en este nivel.
```

Nivel reticular X

-----

```
Número de geometrías              : 205
Número de entradas de índice      : 205
```

2. Determine hasta qué punto los tamaños de celda reticular del índice existente permiten recuperar datos. Evalúe las estadísticas devueltas en el paso anterior.

### Consejo:

- La estadística “Proporción de entradas de índice/geometría” debe ser un valor comprendido entre 1 y 4, preferiblemente valores cercanos a 1.
- El número de entradas de índice por geometría debería ser menor que 10 en el tamaño reticular más grande para evitar el nivel de desbordamiento.

La apariencia de la sección “Nivel reticular X” de la salida de Index Advisor indica que existe un nivel de desbordamiento.

Las estadísticas de índice obtenidas en el paso anterior para COUNTIES\_SHAPE\_IDX indican que los tamaños de retícula (0.5, 0, 0) no son adecuados para los datos de esta columna porque:

- Para el nivel reticular 1, el valor de “Proporción de entradas de índice/geometría” 4.154292 es mayor que la directriz de 4.

La línea “Entradas de índice” tiene los valores 1, 2, 3, 4 y 10, que indican el número de entradas de índice por geometría. Los valores “Absolutos” que se encuentran debajo de cada columna “Entradas de índice” indican el número de geometrías que tienen aquel número específico de entradas de índice. Por ejemplo, la salida del paso anterior muestra que 1519 geometrías tienen 4 entradas de índice. El valor “Absoluto” de 10 entradas de índice es de 695, lo que indica que 695 geometrías tienen entre 5 y 10 entradas de índice.

- La apariencia de la sección “Nivel reticular X” indica que existe un nivel de índice de desbordamiento. Las estadísticas muestran que 205 geometrías tienen más de 10 entradas de índice cada una.
3. Si las estadísticas no son correctas, observe la sección Histograma y las filas apropiadas en las columnas Tamaño de la ventana de consulta y Tamaños de retícula recomendados en la salida de Index Advisor.
- a. Averigüe el tamaño de MBR que tiene el número más grande de geometrías. La sección “Histograma” lista los tamaños de MBR y el número de geometrías que tienen dichos tamaños de MBR. En el siguiente histograma de ejemplo, el número más grande geometrías (437) está en el tamaño de MBR 0.5.

Histograma:

Tamaño de MBR	Número de geometrías
0,040000	1
0,045000	3
0,050000	1
0,055000	3
0,060000	3
0,070000	4
0,075000	3
0,080000	4
0,085000	1
0,090000	2
0,095000	1
0,150000	10
0,200000	9
0,250000	15
0,300000	23
0,350000	83
0,400000	156
0,450000	282
0,500000	437
0,550000	397
0,600000	341
0,650000	246
0,700000	201
0,750000	154
0,800000	120
0,850000	66
0,900000	79
0,950000	59
1,000000	47
1,500000	230
2,000000	89
2,500000	34
3,000000	10
3,500000	5

4,000000	3
5,000000	3
5,500000	2
6,000000	2
6,500000	3
7,000000	2
8,000000	1
15,000000	3
25,000000	2
30,000000	1

b. Vaya a la fila de Tamaño de la ventana de consulta que tenga el valor de 0,5 para obtener los tamaños de retícula recomendados (1,4, 3,5, 14,0).

Tamaño de la ventana de consulta	Tamaños de retícula recomendados			Coste
0.1	0.7,	2.8,	14.0	2.7
0.2	0.7,	2.8,	14.0	2.9
0.5	1.4,	3.5,	14.0	3.5
1	1.4,	3.5,	14.0	4.8
2	1.4,	3.5,	14.0	8.2
5	1.4,	3.5,	14.0	24
10	2.8,	8.4,	21.0	66
20	4.2,	14.7,	37.0	190
50	7.0,	14.0,	70.0	900
100	42.0,	0,	0	2800

4. Compruebe si los tamaños recomendados se ajustan a las directrices del paso 2. Ejecute el mandato gseidx con los tamaños de retícula recomendados:

```
gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY
STATISTICS FOR COLUMN ID_usuario.counties(shape)
USING GRID SIZES (1.4, 3.5, 14.0)
```

Nivel reticular 1  
-----

Tamaño de retícula : 1.4  
Número de geometrías : 3065  
Número de entradas de índice : 5951

Número de celdas reticulares ocupadas : 513  
Proporción de entradas de índice/geometría : 1.941599  
Proporción de geometrías/celda reticular : 5.974659  
Número máximo de geometrías por celda reticular : 42  
Número mínimo de geometrías por celda reticular : 1

Entradas de índice:	1	2	3	4	10
Absoluto :	1180	1377	15	493	0
Porcentaje (%):	38.50	44.93	0.49	16.08	0.00

Nivel reticular 2  
-----

Tamaño reticular : 3.5  
Número de geometrías : 61  
Número de entradas de índice : 143

Número de celdas reticulares ocupadas : 56  
Proporción de entradas de índice/geometría : 2.344262  
Proporción de geometrías/celda reticular : 1.089286  
Número máximo de geometrías por celda reticular : 10  
Número mínimo de geometrías por celda reticular : 1

Entradas de índice:	1	2	3	4	10
---------------------	---	---	---	---	----

```

-----
Absoluto      :      15      28      0      18      0
Porcentaje (%):      24.59  45.90  0.00  29.51  0.00

```

Nivel reticular 3

-----

```

Tamaño de retícula      : 14.0
Número de geometría     : 15
Número de entradas de índice : 28

Número de celdas reticulares ocupadas      : 9
Proporción de entradas de índice/geometría : 1.866667
Proporción de geometrías/celda reticular   : 1.666667
Número máximo de geometrías por celda reticular : 10
Número mínimo de geometrías por celda reticular : 1

```

```

Entradas de índice:  1      2      3      4      10
-----
Absoluto      :      7      5      1      2      0
Porcentaje (%):      46.67  33.33  6.67  13.33  0.00

```

Ahora las estadísticas muestran los valores contenidos en las directrices:

- Los valores de “Proporción de entradas de índice/geometría” son de 1.941599 para el Nivel reticular 1, de 2.344262 para el Nivel reticular 2 y de 1.866667 para el Nivel reticular 3. Todos estos valores están contenidos en el valor de rango de 1 a 4 de las directrices.
  - La ausencia de la sección “Nivel reticular X” indica que ninguna entrada de índice se encuentra en el nivel de desbordamiento.
5. Descarte el índice existente y sustitúyalo por un índice que especifique los tamaños de retícula aconsejados. Para el ejemplo del paso anterior, ejecute las siguientes sentencias de DDL:

```

DROP INDEX ID_usuario.counties_shape_idx;
CREATE INDEX counties_shape_idx ON ID_usuario.counties(shape) EXTEND USING
DB2GSE.SPATIAL_INDEX(1.4,3.5,14.0);

```

## El mandato gseidx

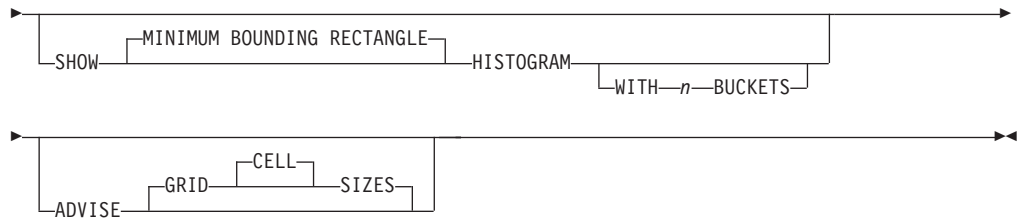
Utilice el mandato gseidx para invocar a Index Advisor para los índices reticulares espaciales.

### Sintaxis

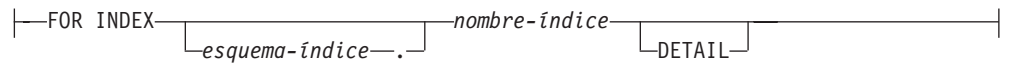
```

▶▶ gseidx CONNECT TO nombre_base_datos [USER id_usuario USING contraseña]
▶ GET GEOMETRY STATISTICS [existing-index | simulated-index]
▶ [ANALYZE número ROWS (1) ONLY] PERCENT

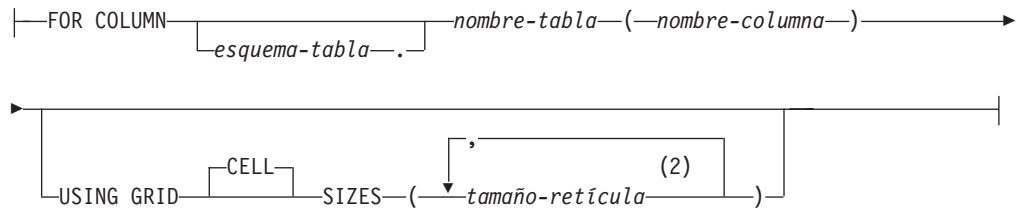
```



**existing-index:**



**simulated-index:**



**Notas:**

- 1 En lugar de la palabra clave PERCENT, puede especificar un símbolo de porcentaje (%).
- 2 Puede especificar tamaños de celda para uno, dos o tres niveles reticulares.

**Parámetros**

**nombre\_base\_datos**

El nombre de la base de datos en la que reside la tabla espacial.

**id\_usuario**

El ID de usuario que tiene autorización DATAACCESS sobre la base de datos en la que reside el índice o la tabla o bien autorización SELECT sobre la tabla. Si inicia la sesión en el entorno de mandatos de DB2 con el ID de usuario del propietario de la base de datos no tendrá que especificar *ID\_usuario* ni *contraseña* con el mandato gseidx.

**contraseña**

La contraseña para el ID de usuario.

**existing-index**

Especifica un índice existente de acuerdo con el cual se recopilan estadísticas.

**esquema-índice**

Nombre del esquema donde está contenido el índice existente.

**nombre-índice**

Nombre no calificado del índice existente.

**DETAIL**

Muestra la información siguiente sobre cada nivel reticular:

- El tamaño de las celdas reticulares

- El número de geometrías indexadas
- El número de entradas de índice
- El número de celdas reticulares que contienen geometrías
- El número promedio de entradas de índice por geometría
- El número promedio de geometrías por celda reticular
- El número de geometrías de la celda que contiene el mayor número de geometrías
- El número de geometrías de la celda que contiene el menor número de geometrías

#### **simulated-index**

Especifica una columna de tabla y un índice simulado para esta columna.

#### **esquema-tabla**

Nombre del esquema donde reside la tabla con la columna para la que está pensado el índice simulado.

#### **nombre-tabla**

Nombre no calificado de la tabla con la columna para la que está pensado el índice simulado.

#### **nombre-columna**

Nombre no calificado de la columna de tabla para la que está pensado el índice simulado.

#### **tamaño-retícula**

Tamaño de las celdas de cada nivel reticular (fino, medio y grueso) de un índice simulado. Debe especificar un tamaño de celda para un nivel como mínimo. Si no desea incluir un nivel, no especifique un tamaño de celda reticular para él o especifique el valor 0.0 (cero) como tamaño de la celda reticular del nivel.

Cuando se especifica el parámetro *tamaño-retícula*, Index Advisor proporciona la misma clase de estadísticas que cuando se especifica la palabra clave *DETAIL* en la cláusula *existing-index*.

#### **ANALYZE número ROWS | PERCENT ONLY**

Reúne estadísticas sobre los datos de un subconjunto de filas de tabla. Si la tabla tiene más de un millón de filas, probablemente deseará utilizar la cláusula *ANALYZE* para disponer de suficiente tiempo de proceso. Especifique el número o porcentaje aproximado de filas que se deben incluir en ese subconjunto.

#### **SHOW MINIMUM BOUNDING RECTANGLE HISTOGRAM**

Visualiza un diagrama que muestra los tamaños de los rectángulos delimitadores mínimos (los MBR) y el número de geometrías cuyos MBR son del mismo tamaño.

#### **WITH n BUCKETS**

Especifica el número de agrupaciones de los MBR de todas las geometrías analizadas. Los MBR pequeños se agrupan junto con otras geometrías pequeñas. Los MBR mayores se agrupan junto con otras geometrías mayores.

Si no especifica este parámetro o especifica 0 cubetas, Index Advisor muestra tamaños de cubeta logarítmicos. Por ejemplo, los tamaños de MBR podrían ser valores logarítmicos tales como 1.0, 2.0, 3.0,... 10.0, 20.0, 30.0,... 100.0, 200.0, 300.0,...

Si especifica un número de compartimentos mayor que 0, Index Advisor muestra valores con el mismo tamaño. Por ejemplo, los valores de MBR podrían ser valores con el mismo tamaño como, por ejemplo, 8,0, 16,0, 24,0,... 320,0, 328,0, 334,0.

El valor por omisión es utilizar compartimentos de tamaño logarítmico.

#### **ADVISE GRID CELL SIZES**

Calcula tamaños de celda reticular próximos al valor óptimo.

#### **Nota sobre el uso**

Si entra el mandato gseidx desde un indicador del sistema operativo, debe escribir todo el mandato en una sola línea.

#### **Ejemplo**

El ejemplo siguiente solicita información detallada sobre un índice reticular existente cuyo nombre es COUNTIES\_SHAPE\_IDX y sugiere tamaños de índice reticular apropiados:

```
gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY  
STATISTICS FOR INDEX ID_usuario.counties_shape_idx DETAIL ADVISE
```

---

## **Utilización de vistas para acceder a columnas espaciales**

Puede definir una vista que hace uso de una columna espacial de la misma forma que define vistas en DB2 para otros tipos de datos.

Si tiene una tabla que tiene una columna espacial y desea que sea utilizada por una vista, utilice las fuentes de información siguientes.



---

## Capítulo 12. Análisis y generación de información espacial

Después de llenar con datos las columnas espaciales, está preparado para consultarlas. Este capítulo:

- Describe los entornos en los que puede someter consultas
- Proporciona ejemplos de los diversos tipos de funciones espaciales que puede invocar en una consulta
- Proporciona directrices sobre el uso de funciones espaciales en combinación con índices espaciales

---

### Entornos para realizar análisis espaciales

Puede realizar análisis espaciales utilizando SQL y funciones espaciales en los entornos de programación siguientes:

- Sentencias de SQL interactivo.

Puede entrar sentencias de SQL interactivo desde el Editor de mandatos de DB2®, la ventana Mandatos de DB2 o el procesador de línea de mandatos de DB2.

- Programas de aplicación en todos los lenguajes soportados por DB2.

---

### Ejemplos del funcionamiento de las funciones espaciales

DB2 Spatial Extender proporciona funciones que realizan diversas operaciones sobre datos espaciales. En términos generales, estas funciones se pueden clasificar según el tipo de operación que realizan. La Tabla 3 indica estas categorías, junto con los ejemplos. El texto que sigue a la Tabla 3 muestra la codificación correspondiente a los ejemplos.

*Tabla 3. Funciones espaciales y operaciones*

Categoría de función	Ejemplo de operación
Devuelve información sobre geometrías determinadas. Realiza comparaciones.	Devolver la extensión, en millas cuadradas, del área de ventas de la Tienda 10. Determinar si el lugar de residencia de un cliente queda dentro del área de ventas de la Tienda 10.
Obtiene nuevas geometrías a partir de otras existentes. Convierte geometrías entre formatos de intercambio de datos.	Obtener el área de ventas de una tienda a partir de su ubicación. Convertir en una geometría la información sobre el cliente expresada en formato GML para que la información se pueda añadir a una base de datos de DB2.

#### Ejemplo 1: Devolver información sobre geometrías específicas

En este ejemplo, la función ST\_Area devuelve un valor numérico que representa el área de ventas de la tienda 10. Esta función devuelve el área utilizando las mismas unidades que el sistema de coordenadas utilizado para definir la ubicación del área.

```
SELECT db2gse.ST_Area(sales_area)
FROM stores
WHERE id = 10
```

El ejemplo siguiente realiza la misma operación que el anterior, pero ST\_Area se ejecuta como método y devuelve el área utilizando la milla cuadrada como unidad.

```
SELECT sales_area..ST_Area('STATUTE MILE')
FROM stores
WHERE id = 10
```

## Ejemplo 2: Realizar comparaciones

En este ejemplo, la función ST\_Within compara las coordenadas de la geometría correspondiente al lugar de residencia de un cliente con las coordenadas de una geometría representativa del área de ventas de la tienda 10. Los datos devueltos por la función indicarán si el lugar de residencia está dentro del área de ventas.

```
SELECT c.first_name, c.last_name, db2gse.ST_Within(c.location, s.sales_area)
FROM customers as c. stores AS s
WHERE s.id = 10
```

## Ejemplo 3: Obtener nuevas geometrías a partir de las existentes

En este ejemplo, la función ST\_Buffer obtiene una geometría representativa del área de ventas de una tienda a partir de una geometría representativa de la ubicación de la tienda.

```
UPDATE stores
SET sales_area = db2gse.ST_Buffer(location, 10, 'KILOMETERS')
WHERE id = 10
```

El ejemplo siguiente realiza la misma operación que el anterior, pero ST\_Buffer se ejecuta como método.

```
UPDATE stores
SET sales_area = location..ST_Buffer(10, 'KILOMETERS')
WHERE id = 10
```

## Ejemplo 4: Convertir geometrías entre varios formatos de intercambio de datos.

En este ejemplo, la información sobre un cliente codificada en formato GML se convierte en una geometría, para que se pueda guardar en una base de datos de DB2.

```
INSERT
INTO c.name,c.phoneNo,c.address
VALUES ( 123, 'Mary Anne', Smith', db2gse.ST_Point('
<gml:Point><gml:coord><gml=X>-130.876</gml:X>
<gml:Y>41.120'</gml:Y></gml:coord></gml:Point>, 1) )
```

---

## Funciones que utilizan índices para optimizar consultas

Un grupo especializado de funciones espaciales, denominado *funciones de comparación* puede mejorar el rendimiento de las consultas aprovechando un índice reticular espacial o un índice Voronoi geodésico (ambos conocidos como *índices espaciales*). Estas funciones comparan dos geometrías entre sí. Si los resultados de la comparación satisfacen determinados criterios, la función devuelve un valor de 1; si los resultados no satisfacen los criterios, la función devuelve un valor de 0. Si la comparación no puede realizarse, la función puede devolver un valor nulo.

Por ejemplo, la función ST\_Overlaps compara dos geometrías que tienen la misma dimensión (por ejemplo, dos cadenas lineales o dos polígonos). Si las geometrías se

solapan parcialmente, y si el espacio abarcado por el solapamiento tiene la misma dimensión que las geometrías, ST\_Overlaps devuelve el valor 1.

La Tabla 4 muestra qué funciones de comparación pueden utilizar un índice reticular espacial y qué funciones pueden utilizar un índice Voronoi geodésico:

Tabla 4. Funciones de comparación que pueden utilizar un índice reticular espacial o un índice Voronoi geodésico

Función de comparación	Puede utilizar un índice reticular espacial	Puede utilizar un índice Voronoi geodésico
EnvelopesIntersect	Sí	Sí
ST_Contains	Sí	Sí
ST_Crosses	Sí	No
ST_Distance	Sí	Sí
ST_EnvIntersects	Sí	Sí
ST_Equals	Sí	No
ST_Intersects	Sí	Sí
ST_MBRIntersects	Sí	Sí
ST_Overlaps	Sí	No
ST_Touches	Sí	No
ST_Within	Sí	Sí

Debido al tiempo y memoria necesarios para ejecutar una función, esta ejecución puede suponer un volumen de proceso considerable. Además, cuando más complejas sean las geometrías que se están comparando, más compleja será la comparación y más tiempo exigirá. Las funciones especializadas listadas anteriormente se pueden ejecutar más rápidamente si pueden utilizar un índice espacial para localizar geometrías. Para que tales funciones puedan utilizar un índice espacial, siga las siguientes normas:

- Especifique la función en una cláusula WHERE. Si está especificada en una cláusula SELECT, HAVING o GROUP BY, no se puede utilizar un índice espacial.
- La función debe ser la expresión situada en el lado izquierdo del predicado.
- El operador utilizado en el predicado que compara el resultado de la función con otra expresión debe ser un signo de igualdad, con una excepción: la función ST\_Distance debe utilizar el operador menor que.
- La expresión situada a la derecha del predicado debe ser la constante 1, excepto en aquellos casos en que ST\_Distance sea la función situada a la derecha.
- La operación debe implicar una búsqueda en una columna espacial en la que se ha definido un índice reticular espacial.

Por ejemplo:

```
SELECT c.name, c.address, c.phone
FROM customers AS c, bank_branches AS b
WHERE db2gse.ST_Distance(c.location, b.location) < 10000
and b.branch_id = 3
```

En la Tabla 5 en la página 98 se muestran las formas correctas e incorrectas de crear consultas espaciales para utilizar un índice espacial.

Tabla 5. Demostración de cómo las funciones espaciales se pueden cumplir y violar normas para utilizar un índice espacial.

Consultas que hacen uso de funciones espaciales	Normas vulneradas
<pre>SELECT * FROM stores AS s WHERE db2gse.ST_Contains(s.sales_zone,     ST_Point(-121.8,37.3, 1)) = 1</pre>	En este ejemplo no se infringe ninguna condición.
<pre>SELECT * FROM stores AS s WHERE db2gse.ST_Length(s.location) &gt; 10</pre>	La función espacial ST_Length no compara geometrías y no puede utilizar un índice espacial.
<pre>SELECT * FROM stores AS s WHERE 1=db2gse.ST_Within(s.location,:BayArea)</pre>	La función debe ser una expresión situada en el lado izquierdo del predicado.
<pre>SELECT * FROM stores AS s WHERE db2gse.ST_Contains(s.sales_zone,     ST_Point(-121.8,37.3, 1)) &lt;&gt; 0</pre>	Las comparaciones de igualdad deben utilizar la constante entera 1.
<pre>SELECT * FROM stores AS s WHERE db2gse.ST_Contains(ST_Polygon     ('polygon((10 10, 10 20, 20 20, 20 10, 10 10))', 1),     ST_Point(-121.8, 37.3, 1)) = 1</pre>	No existe ningún índice espacial en ninguno de los argumentos de la función, por lo que no se puede utilizar ningún índice.

---

## Capítulo 13. Mandatos de DB2 Spatial Extender

Este capítulo explica los mandatos utilizados para configurar DB2 Spatial Extender. También explica cómo utilizar estos mandatos para desarrollar proyectos.

---

### Invocación de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos

Utilice un procesador de línea de mandatos (CLP), denominado db2se, para configurar Spatial Extender y crear proyectos que utilicen datos espaciales. En esta sección se describe cómo utilizar db2se para ejecutar mandatos de DB2 Spatial Extender.

#### Requisitos previos

Para emitir mandatos de db2se, debe estar autorizado para hacerlo. Para saber qué autorización es necesaria para un mandato determinado, consulte en la Tabla 6 en la página 100 el tema del procedimiento almacenado asociado correspondiente al mandato. Por ejemplo, el mandato db2se create\_srs necesita las mismas autorizaciones que el procedimiento almacenado db2.ST\_create\_srs.

**Excepción:** El mandato db2se shape\_info no llama a ningún procedimiento almacenado. En su lugar, muestra información sobre el contenido de los archivos de formas.

Entre los mandatos de db2se desde el indicador del sistema operativo.

Para saber qué submandatos y parámetros puede especificar:

- Escriba db2se o db2se -h; a continuación, pulse Intro. Se visualizará una lista de submandatos de db2se.
- Escriba db2se y un submandato o bien db2se y un submandato seguido de -h. Luego pulse Intro. Se visualizará la sintaxis necesaria del submandato. En esta sintaxis:
  - cada parámetro va precedido por un guión y seguido por un espacio reservado para un valor de parámetro.
  - Los parámetros entre paréntesis son opcionales. Los otros parámetros son necesarios.

**Importante:** Para su comodidad, la sintaxis del mandato se obtiene interactivamente en el monitor; no necesita consultar la sintaxis en ningún otro lugar.

Para emitir un mandato de db2se, escriba db2se. A continuación, escriba un submandato, seguido por los parámetros y los valores de parámetro que el submandato necesite. Finalmente, pulse Intro.

Puede que sea necesario escribir el ID de usuario y la contraseña que dan acceso a la base de datos que ha especificado. Por ejemplo, escriba el ID y la contraseña si desea conectarse a la base de datos como un usuario que no sea el propio. El ID debe ir siempre precedido del parámetro userId y la contraseña debe ir precedida del parámetro pw.

Si no especifica un ID de usuario y una contraseña, se utilizarán por omisión el ID de usuario y la contraseña actuales. Por omisión, los valores especificados no distinguen las mayúsculas de las minúsculas. Si desea realizar esa distinción, encierre los valores entre comillas dobles. Por ejemplo, para especificar el nombre de tabla en minúsculas, mitabla, escriba lo siguiente: "mitabla".

Puede ser necesario anteponer caracteres de escape a las comillas para que no sean interpretadas por el indicador del sistema (shell); por ejemplo, especifique lo siguiente: \`mytable\` Si un valor con distinción de mayúsculas/minúsculas está calificado por otro valor con distinción de mayúsculas/minúsculas, delimite los dos valores individualmente; por ejemplo: "myschema"."mytable" Escriba las series de caracteres entre comillas dobles; por ejemplo: "select \* from newtable"

Cuando se ejecute el mandato db2se, se invocará el procedimiento almacenado que se corresponda con el mandato y se realizará la operación solicitada.

## Visión general de los mandatos de db2se

La tabla siguiente indica qué mandatos de db2se se deben emitir para realizar las tareas que intervienen en la configuración de Spatial Extender y la creación de proyectos que hacen uso de datos espaciales. Esta tabla también proporciona ejemplos de mandatos de db2se y hace referencia a información acerca de autorizaciones y parámetros específicos de los mandatos. A la derecha de la tarea, en la segunda columna, verá un enlace o una referencia a información sobre un procedimiento almacenado. Se llama a este procedimiento almacenado cuando se emite el mandato. La autorización para utilizar el procedimiento almacenado es la misma que la autorización para utilizar el mandato; además, el mandato y el procedimiento almacenado comparten los mismos parámetros. Para saber más sobre la autorización y los significados de los parámetros, consulte la sección identificada por la referencia.

*Tabla 6. Mandatos de db2se clasificados por tarea*

Tarea	Mandato y ejemplo
Crear un sistema de coordenadas.	<p>db2se create_cs</p> <p>Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_create_coordsys.</p> <p>El ejemplo siguiente crea un sistema de coordenadas denominado "misistcoord".</p> <pre>db2se create_cs mydb -coordsysName \<code>"mycoordsys"</code> -definition GEOCS[\<code>"GCS_NORTH_AMERICAN_1983"</code>], DATUM[\<code>"D_North_American_1983"</code>], SPHEROID[\<code>"GRS_1980"</code>,6387137,298.257222101]], PRIMEM[\<code>"Greenwich"</code>,0],UNIT[\<code>"Degree"</code>, 0.0174532925199432955]]</pre>
Crear un sistema de referencia espacial.	<p>db2se create_srs</p> <p>Los parámetros específicos del mandato son los mismos que los del procedimiento almacenado. No es necesaria ninguna autorización.</p> <p>El ejemplo siguiente crea un sistema de referencia espacial denominado "misrs".</p> <pre>db2se create_srs mydb -srsName \<code>"mysrs"</code> -srsID 100 -xScale 10 -coordsysName \<code>"GCS_North_American_1983"</code></pre>

Tabla 6. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo
Descartar un sistema de referencia espacial.	<p>db2se drop_srs</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente descarta un sistema de referencia espacial denominado "misrs".</p> <pre>db2se drop_srs mydb -srsName \"mysrs\"</pre>
Suprimir una definición de sistema de coordenadas.	<p>db2se drop_cs</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente descarta un sistema de coordenadas denominado "misistcoord".</p> <pre>db2se drop_cs mydb -coordsysName \"mycoordsys\"</pre>
Inhabilitar una configuración para geocodificar datos automáticamente.	<p>db2se disable_autogc</p> <p>Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_disable_autogeocoding.</p> <p>El ejemplo siguiente inhabilita la geocodificación automática para una columna geocodificada denominada MYCOLUMN en la tabla MYTABLE.</p> <pre>db2se disable_autogc mydb -tableName \"mytable\" -columnName \"mycolumn\"</pre>
Habilitar una base de datos para operaciones espaciales.	<p>db2se enable_db</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>Asegúrese de que dispone de un espacio de tablas temporal del sistema con un tamaño de página de 8 KB o más y un tamaño mínimo de 500 páginas. Si no dispone de un espacio de tablas con estas características, consulte la sección sobre "creación de espacios de tablas temporales" en <i>Database Administration Concepts and Configuration Reference</i> para obtener más información sobre cómo crearlo. Es un requisito para ejecutar satisfactoriamente el mandato db2se enable_db.</p> <p>El ejemplo siguiente habilita una base de datos denominada MIBD para operaciones espaciales.</p> <pre>db2se enable_db mydb</pre>
Exportar datos a archivos de formas.	<p>db2se export_shape</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente exporta una columna espacial denominada MICOLUMNA y su tabla asociada, MITABLA, a un archivo de formas denominado miarchivo_formas.</p> <pre>db2se export_shape mi_bd -fileName /home/myaccount/miarchivo_formas -selectStatement "select * from mytable"</pre>

Tabla 6. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo
Importar archivos de formas.	<p>db2se import_shape</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El mandato siguiente importa un archivo de formas denominado miarchivo a una tabla denominada MYTABLE. Durante la importación, los datos espaciales de miarchivo se insertan en una columna MITABLA denominada MICOLUMNA.</p> <pre>db2se import_shape mydb -fileName \"myfile\" -srsName NAD83_SRS_1 -tableName \"mytable\" -spatialColumnName \"mycolumn\"</pre>
Registrar un geocodificador.	<p>db2se register_gc</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente registra un geocodificador denominado “mygeocoder”, que es utilizado por una función denominada “myschema.myfunction”.</p> <pre>db2se register_gc mydb -geocoderName \"mygeocoder\" -functionSchema \"myschema\" -functionName \"myfunction\" -defaultParameterValues \"1, 'string',,cast(null as varchar(50))\" -vendor myvendor -description \"myvendor geocoder returning well-known text\"</pre>
Registrar una columna espacial.	<p>db2se register_spatial_column</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente registra una columna espacial denominada MYCOLUMN en la tabla MYTABLE, con el sistema de referencia espacial “USA_SRS_1”.</p> <pre>db2se register_spatial_column mi_bd -tableName \"mytable\" -columnName \"mycolumn\" -srsName USA_SRS_1</pre>
Eliminar los recursos que habilitan una base de datos para operaciones espaciales.	<p>db2se disable_db</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente elimina los recursos que habilitan la base de datos MIBD para operaciones espaciales.</p> <pre>db2se disable_db mydb</pre>
Eliminar una configuración para operaciones de geocodificación.	<p>db2se remove_gc_setup</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente elimina una configuración para las operaciones de geocodificación que se aplican a una columna espacial denominada MICOLUMNA de la tabla MITABLA.</p> <pre>db2se remove_geocoding_setup mi_bd -tableName \"mytable\" -columnName \"mycolumn\"</pre>



Tabla 6. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo
Ejecutar un geocodificador en la modalidad de proceso por lotes.	<p>db2se run_gc</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente ejecuta un geocodificador en modalidad de proceso por lotes para llenar una columna denominada MYCOLUMN de una tabla denominada MYTABLE.</p> <pre>db2se run_gc mydb -tableName \"mytable\" -columnName \"mycolumn\"</pre>
Configurar un geocodificador para que se ejecute automáticamente.	<p>db2se enable_autogeocoding</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente configura una geocodificación automática para una columna denominada MICOLUMNA en la tabla MITABLA</p> <pre>db2se enable_autogeocoding mi_bd -tableName \"mytable\" -columnName \"mycolumn\"</pre>
Configurar operaciones de geocodificación.	<p>db2se setup_gc</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente configura operaciones de geocodificación para llenar una columna espacial denominada MICOLUMNA en la tabla MITABLA.</p> <pre>db2se setup_gc mydb -tableName \"mytable\" -columnName \"mycolumn\" -geocoderName \"db2se_USA_GEOCODER\" -parameterValues \"address,city,state,zip,2,90,70,20,1.1,'meter',4..\" -autogeocodingColumns address,city,state,zip commitScope 10</pre>
Mostrar información acerca de un archivo de formas y su contenido.	<p>db2se shape_info</p> <p>Para utilizar este mandato, debe:</p> <ul style="list-style-type: none"> <li>• Tener permiso para leer el archivo al que hace referencia el mandato.</li> <li>• Poder conectarse a la base de datos que contiene este archivo (si utiliza el parámetro <i>-database</i>, que especifica que el sistema busca en la base de datos definida sistemas de coordenadas y sistemas de referencia espacial compatibles).</li> </ul> <p>El ejemplo siguiente muestra información acerca del archivo de formas denominado myfile, que reside en el directorio actual.</p> <pre>db2se shape_info -fileName myfile</pre> <p>El ejemplo siguiente muestra información acerca de un archivo de formas de ejemplo para UNIX denominado offices. El parámetro <i>-database</i> busca todos los sistemas de referencia espacial y los sistemas de coordenadas compatibles en la base de datos definida (en este caso, MIBD).</p> <pre>db2se shape_info -fileName ~/sqllib/samples/extenders/spatial/data/offices -database miBD</pre>

Tabla 6. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo
Deshacer el registro de un codificador.	<p>db2se unregister_gc</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente cancela el registro de un geocodificador denominado "mygeocoder".</p> <pre>db2se unregister_gc mydb -geocoderName \mygeocoder\</pre>
Cancelar el registro de una columna espacial.	<p>db2se unregister_spatial_column</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente cancela el registro de una columna espacial denominada mycolumn en la tabla mytable.</p> <pre>db2se unregister_spatial_column mydb -tableName \mytable\ -columnName \mycolumn\</pre>
Actualizar una definición de sistema de coordenadas.	<p>db2se alter_cs</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente actualiza la definición de un sistema de coordenadas denominado "mycoordsys" con un nuevo nombre de organización.</p> <pre>db2se alter_cs mydb -coordsysName \mycoordsys\ -organization myNeworganization -tableName \mytable\</pre>
Actualizar una definición de sistema de referencia espacial.	<p>db2se alter_srs</p> <p>Los parámetros específicos del mandato y las autorizaciones necesarias son los mismos que los del procedimiento almacenado.</p> <p>El ejemplo siguiente modifica un sistema de referencia espacial denominado "mirs" con un xOffset y una descripción diferentes.</p> <pre>db2se alter_srs mydb -srsName \mysrs\ -xoffset 35 -description "Este es mi propio sistema de referencia espacial."</pre>

## Mandato db2se upgrade

El mandato db2se upgrade actualiza una base de datos espacial de la versión 8, versión 9.1 o versión 9.5 a la versión 9.7.

Este mandato puede eliminar y reconstruir índices espaciales para completar la actualización de la base de datos, lo cual puede exigir mucho tiempo dependiendo del tamaño de las tablas. Por ejemplo, se eliminan y reconstruyen los índices si los datos se trasladan desde una instancia de 32 bits a una instancia de 64 bits, o si realiza la actualización desde DB2 UDB Versión 8 FixPack 6 o anterior.

**Consejo:** Ejecute el mandato db2se upgrade con la opción `-force 0` y especifique un archivo de mensajes para determinar qué índices se deben actualizar sin realizar un proceso de actualización adicional.

## Autorización

Autorización DBADM y DATAACCESS sobre la base de datos espacial que desea actualizar.

## Sintaxis del mandato

### Mandato db2se upgrade

```
db2se upgrade nombre_base_datos  
-userId ID_usuario -pw contraseña  
-tableCreationParameters parámetros_creación_tablas  
-force valor_force -messagesFile nombre_archivo_mensajes
```

## Parámetros del mandato

Donde:

### nombre\_base\_datos

Nombre de la base de datos que desea actualizar.

### -userId user\_id

ID de usuario de base de datos que tiene la autorización DATAACCESS sobre la base de datos que se está actualizando.

### -pw password

Contraseña de usuario.

### -tableCreationParameters table\_creation\_parameters

Parámetros que se deben utilizar en la creación de las tablas de catálogo de Spatial Extender.

### -force force\_value

- 0: valor por omisión. Intenta realizar la actualización, pero se detiene si cualquier objeto definido por el usuario, como vistas, funciones, activadores o índices espaciales, hacen referencia a objetos de Spatial Extender.
- 1: guarda y restaura automáticamente objetos definidos por la aplicación. Guarda y restaura índices espaciales si es necesario.
- 2: guarda y restaura automáticamente objetos definidos por la aplicación. Guarda información de índices espaciales, pero no restaura automáticamente índices espaciales.

### -messagesFile messages\_filename

Nombre del archivo que contiene el informe de las acciones de actualización. El nombre de archivo que especifique debe ser un nombre de archivo calificado al completo en el servidor.

**Consejo:** Especifique este parámetro para ayudarle en la resolución de problemas de actualización.

**Restricción:** No puede especificar un archivo existente.

## Notas sobre el uso

El mandato db2se upgrade verifica diversas condiciones y devuelve uno o varios errores si no se cumple alguna de las condiciones siguientes:

- La base de datos no está actualmente habilitada para operaciones espaciales.
- El nombre de la base de datos no es válido.
- Existen otras conexiones a la base de datos. No se puede continuar con la actualización.
- El catálogo espacial no es consistente.
- El usuario no está autorizado.
- La contraseña no es válida.
- No se han podido actualizar algunos objetos definidos por el usuario.

Asegúrese de que dispone de un espacio de tablas temporal del sistema con un tamaño de página de 8 KB o más y un tamaño mínimo de 500 páginas. Si no dispone de un espacio de tablas con estas características, consulte la sección sobre “creación de espacios de tablas temporales” en *Database Administration Concepts and Configuration Reference* para obtener más información sobre cómo crearlo. Es un requisito para ejecutar satisfactoriamente el mandato db2se upgrade.

---

## Mandato db2se migrate

El mandato db2se migrate migra una base de datos espacial a la versión 9.7. Este mandato está en desuso y se eliminará en un futuro release. Utilice en su lugar el mandato db2se upgrade.

Este mandato puede eliminar y reconstruir índices espaciales para completar la migración, lo cual puede exigir mucho tiempo dependiendo del tamaño de las tablas. Por ejemplo, se eliminan y reconstruyen los índices si los datos se trasladan desde una instancia de 32 bits a una instancia de 64 bits, o si realiza la actualización desde DB2 Versión 8 FixPack 6 o anterior.

**Consejo:** Ejecute el mandato db2se migrate con la opción `-force 0` y especifique un archivo de mensajes para determinar qué índices se deben migrar sin realizar un proceso de migración adicional.

### Autorización

Autorización SYSADM o DBADM sobre la base de datos espacial que desea migrar.

### Sintaxis del mandato

#### Mandato db2se migrate

```
►►—db2se migrate—nombre_base_datos—►►
►
└── -userId—ID_usuario— -pw—contraseña—┘
►
└── -tableCreationParameters—parámetros_creación_tablas—┘
```



## Parámetros del mandato

Donde:

**nombre\_base\_datos**

Nombre de la base de datos que desea migrar.

**-userId ID\_usuario**

ID de usuario de base de datos que tiene la autorización SYSADM o DBADM sobre la base de datos que se está migrando.

**-pw contraseña**

Contraseña de usuario.

**-tableCreationParameters parámetros\_creación\_tablas**

Parámetros que se deben utilizar en la creación de las tablas de catálogo de Spatial Extender.

**-force valor\_forzar**

- 0: valor por omisión. Intenta realizar la migración de la base de datos, pero se detiene si cualquier objeto definido por la aplicación, tal como vistas, funciones, activadores o índices espaciales se han basado en objetos de Spatial Extender.
- 1: guarda y restaura automáticamente objetos definidos por la aplicación. Guarda y restaura índices espaciales si es necesario.
- 2: guarda y restaura automáticamente objetos definidos por la aplicación. Guarda información de índices espaciales, pero no restaura automáticamente índices espaciales.

**-messagesFile nombre\_archivo\_mensajes**

Nombre del archivo que contiene el informe de las acciones de migración. El nombre de archivo que especifique debe ser un nombre de archivo calificado al completo en el servidor.

**Consejo:** Especifique este parámetro para ayudarle en la resolución de problemas de migración.

**Restricción:** No puede especificar un archivo existente.

Puede que reciba uno o más de los siguientes errores durante la migración:

- La base de datos no está actualmente habilitada para operaciones espaciales.
- El nombre de la base de datos no es válido.
- Existen otras conexiones a la base de datos. No se puede ejecutar.
- El catálogo espacial no es consistente.
- El usuario no está autorizado.
- La contraseña no es válida.
- No se han podido migrar algunos objetos de usuario.

---

## Mandato db2se restore\_indexes

Restaura los índices espaciales que ha guardado previamente emitiendo el mandato db2se save\_indexes a una base de datos espacial.

## Autorización

Autorización DBADM y DATAACCESS sobre la base de datos espacial.

## Sintaxis del mandato

### Mandato db2se restore\_indexes

```
db2se restore_indexes nombre_base_datos  
-userId ID_usuario -pw contraseña -messagesFile nombre_archivo_mensajes
```

## Parámetros de mandatos

### nombre\_base\_datos

Nombre de la base de datos que desea migrar.

### -userId ID\_usuario

ID de usuario de base de datos que tiene la autorización SYSADM o DBADM sobre la base de datos que se está migrando.

### -pw contraseña

Contraseña de usuario.

### -messagesFile nombre\_archivo\_mensajes

Nombre del archivo que contiene el informe de las acciones de migración. El nombre de archivo que especifique debe ser un nombre de archivo calificado al completo en el servidor.

---

## Mandato db2se save\_indexes

Guarda los índices espaciales definidos en una base de datos espacial.

## Autorización

Autorización DBADM y DATAACCESS sobre la base de datos espacial.

## Sintaxis del mandato

### Mandato db2se save\_indexes

```
db2se save_indexes nombre_base_datos  
-userId ID_usuario -pw contraseña -messagesFile nombre_archivo_mensajes
```

## Parámetros de mandatos

### nombre\_base\_datos

Nombre de la base de datos que desea migrar.

### -userId ID\_usuario

ID de usuario de base de datos que tiene la autorización SYSADM o DBADM sobre la base de datos que se está migrando.

### -pw contraseña

Contraseña de usuario.

**-messagesFile nombre\_archivo\_mensajes**

Nombre del archivo que contiene el informe de las acciones de migración.  
El nombre de archivo que especifique debe ser un nombre de archivo calificado al completo en el servidor.





---

## Capítulo 14. Escritura de aplicaciones y utilización del programa de ejemplo

Este capítulo explica cómo escribir aplicaciones de Spatial Extender.

---

### Cómo incluir el archivo de cabecera de DB2 Spatial Extender en aplicaciones espaciales

DB2 Spatial Extender proporciona un archivo de cabecera que define constantes que se pueden utilizar con las funciones y los procedimientos almacenados de DB2 Spatial Extender.

#### Recomendación:

Si tiene la intención de llamar a procedimientos almacenados o funciones de DB2 Spatial Extender desde programas C o C++, incluya este archivo de cabecera en las aplicaciones espaciales.

Para realizar esta tarea... :

1. Asegúrese de que las aplicaciones de DB2 Spatial Extender pueden utilizar las definiciones necesarias en este archivo de cabecera.
  - a. Incluya el archivo de cabecera de DB2 Spatial Extender en el programa de aplicación. El archivo de cabecera tiene el nombre siguiente:  
db2gse.h  
El archivo de cabecera está situado en el directorio *db2path*/include, donde *db2path* es el directorio de instalación donde está instalado el sistema de base de datos de DB2.
  - b. Asegúrese de que la vía de acceso del directorio include esté especificada en el *makefile* con la opción de compilación.
2. Si está creando aplicaciones Windows de 64 bits en un sistema Windows de 32 bits, cambie el parámetro `DB2_LIBS` del archivo `samples/extenders/spatial/makefile.nt` para alojar aplicaciones de 64 bits. Los cambios necesarios se resaltan a continuación:

```
DB2_LIBS = $(DB2_DIR)\lib\Win64\db2api.lib
```

---

### Cómo llamar a los procedimientos almacenados de DB2 Spatial Extender desde una aplicación

Si piensa escribir programas de aplicación que llamen a cualquiera de los procedimientos almacenados de DB2 Spatial Extender, utilice la sentencia `CALL` de SQL y especifique el nombre del procedimiento almacenado.

#### Acerca de esta tarea

Los procedimientos almacenados de DB2 Spatial Extender se crean cuando el usuario habilita la base de datos para operaciones espaciales.

#### Procedimiento

Para realizar esta tarea:

1. Invoque el procedimiento almacenado `ST_enable_db` stored procedure para habilitar una base de datos para operaciones espaciales.  
Especifique el nombre de procedimiento almacenado de la forma siguiente:  
`CALL db2gse!ST_enable_db`  
`db2gse!` en esta llamada representa el nombre de la biblioteca de DB2 Spatial Extender. El procedimiento almacenado `ST_enable_db` es el único en el que necesita incluir un signo de exclamación en la llamada (es decir, `db2gse!`).
2. Invoque otros procedimientos almacenados de DB2 Spatial Extender.  
Especifique el nombre de procedimiento almacenado de la siguiente forma, donde `db2gse` es el nombre de esquema de todos los procedimientos almacenados de DB2 Spatial Extender y `spatial_procedure_name` es el nombre del procedimiento almacenado. No incluya un signo de exclamación en la llamada.  
`CALL db2gse.spatial_procedure_name`  
Los procedimientos almacenados de DB2 Spatial Extender en muestran en la tabla siguiente.

Tabla 7. Procedimientos almacenados de DB2 Spatial Extender

Procedimiento almacenado	Descripción
<code>ST_alter_coordsys</code>	Actualiza un atributo de un sistema de coordenadas en la base de datos.
<code>ST_alter_srs</code>	Actualiza un atributo de un sistema de referencia espacial en la base de datos.
<code>ST_create_coordsys</code>	Crea un sistema de coordenadas en la base de datos.
<code>ST_create_srs</code>	Crea un sistema de referencia espacial en la base de datos.
<code>ST_disable_autogeocoding</code>	Especifica que DB2 Spatial Extender dejará de sincronizar una columna geocodificada con sus columnas de geocodificación asociadas.
<code>ST_disable_db</code>	Elimina recursos que permiten a DB2 Spatial Extender almacenar datos espaciales y dar soporte a operaciones que se realizan sobre estos datos.
<code>ST_drop_coordsys</code>	Suprime un sistema de coordenadas de la base de datos.
<code>ST_drop_srs</code>	Suprime un sistema de referencia espacial de la base de datos.
<code>ST_enable_autogeocoding</code>	Especifica que DB2 Spatial Extender sincronizará una columna geocodificada con sus columnas de geocodificación asociadas.
<code>ST_enable_db</code>	Proporciona una base de datos con los recursos que necesita para almacenar datos espaciales y dar soporte a operaciones.
<code>ST_export_shape</code>	Exporta los datos seleccionados en la base de datos a un archivo de formas.
<code>ST_import_shape</code>	Importa un archivo de formas a una base de datos.
<code>ST_register_geocoder</code>	Registra un geocodificador que no sea <code>DB2SE_USA_GEOCODER</code> , que forma parte del producto DB2 Spatial Extender.

Tabla 7. Procedimientos almacenados de DB2 Spatial Extender (continuación)

Procedimiento almacenado	Descripción
ST_register_spatial_column	Registra una columna espacial y asocia un sistema de referencia espacial a la misma.
ST_remove_geocoding_setup	Elimina toda la información de configuración de geocodificación para la columna geocodificada.
ST_run_geocoding	Ejecuta un geocodificador en modalidad de proceso por lotes.
ST_setup_geocoding	Asocia una columna que se va a geocodificar a un geocodificador y configura los valores de los parámetros de configuración correspondientes.
ST_unregister_geocoder	Deshace el registro de un geocodificador que no sea DB2SE_USA_GEOCODER.
ST_unregister_spatial_column	Elimina el registro de una columna espacial.

## Programa de ejemplo de DB2 Spatial Extender

El programa de ejemplo de DB2<sup>®</sup> Spatial Extender, runGseDemo, tiene dos finalidades. Puede utilizar el programa de ejemplo para familiarizarse con la programación de aplicaciones de DB2 Spatial Extender y para verificar la instalación de DB2 Spatial Extender.

- En UNIX<sup>®</sup>, puede localizar el programa runGseDemo en la siguiente vía de acceso:

```
$HOME/sql1lib/samples/extenders/spatial
```

donde \$HOME es el directorio inicial del propietario.

- En Windows<sup>®</sup>, puede localizar el programa runGseDemo en la vía de acceso siguiente:

```
c:\Archivos de programa\IBM\sql1lib\samples\extenders\spatial
```

donde c:\Archivos de programa\IBM\sql1lib es el directorio en el que ha instalado DB2 Spatial Extender.

El programa de ejemplo DB2 Spatial Extender runGseDemo hace más fácil la programación de aplicaciones. Mediante este programa de ejemplo, puede habilitar una base de datos para operaciones espaciales y realizará análisis espaciales sobre los datos de esa base de datos. Esta base de datos contendrá tablas con información ficticia sobre clientes y sobre zonas con riesgo de inundación. A partir de esta información puede experimentar con Spatial Extender y determinar qué clientes corren el riesgo de sufrir daños a causa de una inundación.

Mediante el programa de ejemplo, puede:

- Ver los pasos habituales necesarios para crear y mantener una base de datos habilitada para operaciones espaciales.
- Conocer cómo invocar a procedimientos almacenados espaciales desde un programa de aplicación.
- Cortar y pegar código de ejemplo en sus propias aplicaciones.

Utilice el programa de ejemplo siguiente para codificar tareas para DB2 Spatial Extender. Por ejemplo, suponga que escribe una aplicación que utiliza la interfaz de base de datos para invocar a procedimientos almacenados de DB2 Spatial Extender. Desde el programa de ejemplo, puede copiar código para personalizar su aplicación. Si no conoce los pasos de programación de DB2 Spatial Extender, puede ejecutar el programa de ejemplo para ver cada paso con detalle. Para obtener instrucciones sobre la ejecución del programa de ejemplo, consulte el apartado “Tareas relacionadas” al final de este tema.

La tabla siguiente describe cada paso en el programa de ejemplo. En cada paso realizará una acción y, en muchos casos, deshará esa acción. Por ejemplo, en el primer paso, habilitará la base de datos espacial y luego la inhabilitará. De esta forma, se familiarizará con muchos de los procedimientos almacenados de Spatial Extender.

*Tabla 8. Pasos del programa de ejemplo de DB2 Spatial Extender*

Pasos	Acción y descripción
Habilitar o inhabilitar la base de datos espacial	<ul style="list-style-type: none"> <li data-bbox="727 724 1122 749">• Habilitar la base de datos espacial Este es el primer paso necesario para utilizar DB2 Spatial Extender. Una base de datos que ha sido habilitada para operaciones espaciales tiene un conjunto de tipos espaciales, un conjunto de funciones espaciales, un conjunto de predicados espaciales, nuevos tipos de índice y un conjunto de tablas y vistas espaciales de catálogo.</li> <li data-bbox="727 947 1427 1213">• Inhabilitar la base de datos espacial Este paso se suele ejecutar cuando se han habilitado características espaciales para una base de datos incorrecta o cuando ya no es necesario realizar operaciones espaciales en la base de datos en cuestión. Cuando inhabilita una base de datos espacial, elimina el conjunto de tipos espaciales, el conjunto de funciones espaciales, el conjunto de predicados espaciales, los nuevos tipos de índice y el conjunto de tablas y vistas de catálogo espaciales asociados a esa base de datos.</li> </ul>
Crear o descartar un sistema de coordenadas	<ul style="list-style-type: none"> <li data-bbox="727 1224 1122 1287">• Habilitar la base de datos espacial Igual que en el caso anterior.</li> <li data-bbox="727 1297 1427 1419">• Crear un sistema de coordenadas llamado NORTH_AMERICAN Este paso crea un nuevo sistema de coordenadas en la base de datos.</li> <li data-bbox="727 1430 1427 1551">• Descartar el sistema de coordenadas llamado NORTH_AMERICAN Este paso descarta el sistema de coordenadas NORTH_AMERICAN de la base de datos.</li> <li data-bbox="727 1562 1427 1688">• Crear un sistema de coordenadas llamado KY_STATE_PLANE Este paso crea un nuevo sistema de coordenadas, KY_STATE_PLANE, que será utilizado por el sistema de referencia espacial creado en el paso siguiente.</li> </ul>

Tabla 8. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Crear o descartar un sistema de referencia espacial	<ul style="list-style-type: none"> <li>• Crear un sistema de referencia espacial llamado SRSDEMO1 Este paso define un nuevo sistema de referencia espacial (spatial reference system, SRS) que se utiliza para interpretar las coordenadas. El SRS comprende datos sobre geometrías en un formato que se puede guardar en una columna de una base de datos habilitada para operaciones espaciales. Una vez registrado el SRS en una columna espacial determinada, las coordenadas aplicables a esa columna espacial se pueden guardar en la columna correspondiente de la tabla CUSTOMERS.</li> <li>• Descartar el SRS llamado SRSDEMO1 Este paso se realiza cuando el SRS ya no es necesario en la base de datos. Cuando descarta un SRS, elimina la definición del SRS contenida en la base de datos.</li> </ul>
Crear y llenar las tablas espaciales	<ul style="list-style-type: none"> <li>• Crear el SRS llamado KY_STATE_SRS</li> <li>• Crear la tabla CUSTOMERS</li> <li>• Llenar la tabla CUSTOMERS La tabla CUSTOMERS representa los datos comerciales que se han almacenado en la base de datos durante varios años.</li> <li>• Alterar la tabla CUSTOMERS añadiendo la columna LOCATION La sentencia ALTER TABLE añade una nueva columna (LOCATION) de tipo ST_Point. Esta columna se llenará geocodificando las columnas de dirección en un paso posterior.</li> <li>• Crear la tabla OFFICES La tabla OFFICES representa, entre otros datos, la zona de ventas de cada oficina de una compañía de seguros. En un paso posterior, la tabla completa se llenará con los datos de atributos procedentes de una base de datos no perteneciente a DB2. Este paso posterior supone importar datos de atributos a la tabla OFFICES desde un archivo de formas.</li> </ul>

Tabla 8. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Llenar las columnas	<ul style="list-style-type: none"> <li data-bbox="727 258 1427 352">• Geocodificar los datos de direcciones para la columna LOCATION de la tabla CUSTOMERS con el geocodificador denominado KY_STATE_GC Este paso realiza una geocodificación espacial por lotes mediante la invocación del programa geocodificador. La geocodificación por lotes se suele realizar cuando es necesario geocodificar o volver a geocodificar una parte significativa de la tabla.</li> <li data-bbox="727 510 1427 720">• Cargar la tabla OFFICES anteriormente creada desde el archivo de formas utilizando el sistema de referencia espacial KY_STATE_SRS Este paso carga la tabla OFFICES con datos espaciales existentes en forma de un archivo de formas. Debido a que la tabla OFFICES ya existe, el programa de carga añadirá los nuevos registros a una tabla existente.</li> <li data-bbox="727 730 1427 940">• Crear y cargar la tabla FLOODZONES desde el archivo de formas utilizando el sistema de referencia espacial KY_STATE_SRS Este paso carga la tabla FLOODZONES con datos existentes en forma de un archivo de formas. Debido a que la tabla no existe, el programa de carga creará la tabla antes de cargar los datos.</li> <li data-bbox="727 951 1427 1003">• Crear y cargar la tabla REGIONS desde el archivo de formas utilizando el sistema de referencia espacial KY_STATE_SRS</li> </ul>
Registrar o desregistrar el geocodificador	<ul style="list-style-type: none"> <li data-bbox="727 1014 1276 1045">• Registrar el geocodificador llamado SAMPLEGC</li> <li data-bbox="727 1056 1308 1087">• Desregistrar el geocodificador llamado SAMPLEGC</li> <li data-bbox="727 1098 1211 1129">• Registrar el geocodificador KY_STATE_GC</li> </ul> <p data-bbox="727 1150 1427 1234">Estos pasos registran y desregistran el geocodificador llamado SAMPLEGC y luego crean un nuevo geocodificador, KY_STATE_GC, para utilizarlo en el programa de ejemplo.</p>
Crear índices espaciales	<ul style="list-style-type: none"> <li data-bbox="727 1245 1427 1297">• Crear el índice reticular espacial para la columna LOCATION de la tabla CUSTOMERS</li> <li data-bbox="727 1308 1325 1360">• Descartar el índice reticular espacial para la columna LOCATION de la tabla CUSTOMERS</li> <li data-bbox="727 1371 1427 1423">• Crear el índice reticular espacial para la columna LOCATION de la tabla CUSTOMERS</li> <li data-bbox="727 1434 1427 1486">• Crear el índice reticular espacial para la columna LOCATION de la tabla OFFICES</li> <li data-bbox="727 1497 1427 1549">• Crear el índice reticular espacial para la columna LOCATION de la tabla FLOODZONES</li> <li data-bbox="727 1560 1427 1612">• Crear el índice reticular espacial para la columna LOCATION de la tabla REGIONS</li> </ul> <p data-bbox="727 1654 1427 1715">Estos pasos crean el índice reticular espacial para las tablas CUSTOMERS, OFFICES, FLOODZONES y REGIONS.</p>

Tabla 8. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Habilitar la geocodificación automática.	<ul style="list-style-type: none"> <li>• Configurar la geocodificación para la columna LOCATION de la tabla CUSTOMERS con el geocodificador KY_STATE_GC Este paso asocia la columna LOCATION de la tabla CUSTOMERS con el geocodificador KY_STATE_GC y configura los correspondientes valores de parámetros de geocodificación.</li> <li>• Habilitar la geocodificación automática para la columna LOCATION de la tabla CUSTOMERS Este paso activa la invocación automática del geocodificador. La geocodificación automática hace que las columnas LOCATION, STREET, CITY, STATE y ZIP de la tabla CUSTOMERS se sincronicen entre sí para operaciones subsiguientes de inserción y actualización.</li> </ul>
Realizar operaciones de inserción, actualización y supresión sobre la tabla CUSTOMERS	<ul style="list-style-type: none"> <li>• Insertar algunos registros con una calle diferente</li> <li>• Actualizar algunos registros con una nueva dirección</li> <li>• Suprimir todos los registros de la tabla</li> </ul> <p>Estos pasos muestran operaciones de inserción, actualización y supresión sobre las columnas STREET, CITY, STATE y ZIP de la tabla CUSTOMERS. Una vez habilitada la geocodificación automática, los datos que se insertan o actualizan en estas columnas se geocodifican automáticamente en la columna LOCATION. Este proceso se habilitó en el paso anterior.</p>
Inhabilitar la geocodificación automática	<ul style="list-style-type: none"> <li>• Inhabilitar la geocodificación automática para la columna LOCATION de la tabla CUSTOMERS</li> <li>• Eliminar la definición de geocodificación para la columna LOCATION de la tabla CUSTOMERS</li> <li>• Descartar el índice espacial para la columna LOCATION de la tabla CUSTOMERS</li> </ul> <p>Estos pasos inhabilitan la invocación automática del geocodificador y el índice espacial como preparación para el paso siguiente. El paso siguiente vuelve a geocodificar la tabla CUSTOMERS completa.</p> <p><b>Recomendación:</b> si carga un volumen grande de datos espaciales, descarte el índice espacial antes de cargar los datos, y luego vuelva a crearlo una vez cargados los datos.</p>

Tabla 8. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Volver a geocodificar la tabla CUSTOMERS	<ul style="list-style-type: none"> <li>• Volver a geocodificar la columna LOCATION de la tabla CUSTOMERS con un nivel de precisión menor: 90% en lugar de 100%</li> <li>• Volver a crear el índice espacial para la columna LOCATION de la tabla CUSTOMERS</li> <li>• Habilitar de nuevo la geocodificación automática con un nivel de precisión menor: 90% en lugar de 100%</li> </ul> <p data-bbox="727 510 1425 940">Estos pasos ejecutan el geocodificador en la modalidad de proceso por lotes, crean de nuevo el índice espacial y vuelven a habilitar la geocodificación automática con un nuevo nivel de precisión. Esta acción es recomendable cuando se observa una tasa de errores elevada en el proceso de geocodificación. Si el nivel de precisión está establecido en 100%, la geocodificación de una dirección puede fallar debido a que no puede encontrar una dirección coincidente en los datos de referencia. Mediante la disminución del nivel de precisión, el geocodificador puede tener más probabilidad de encontrar datos coincidentes. Después de volver a geocodificar la tabla en la modalidad de proceso por lotes, se habilita de nuevo la geocodificación automática y se vuelve a crear el índice espacial. Esto facilita el mantenimiento incremental del índice espacial y de la columna espacial para operaciones posteriores de inserción y actualización.</p>
Crear una vista y registrar la columna espacial en la vista	<ul style="list-style-type: none"> <li>• Crear una vista denominada HIGHRISKCUSTOMERS basada en la unión de las tablas CUSTOMERS y FLOODZONES</li> <li>• Registrar la columna espacial de la vista</li> </ul> <p data-bbox="727 1077 1377 1098">Estos pasos crean una vista y registran su columna espacial.</p>



Tabla 8. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Realizar el análisis espacial	<ul style="list-style-type: none"> <li>• Encontrar el número de clientes atendidos en cada región (ST_Within)</li> <li>• Para las oficinas y clientes de la misma región, encontrar el número de clientes que están dentro de una distancia determinada respecto a cada oficina (ST_Within, ST_Distance)</li> <li>• Para cada región, encontrar los ingresos medios y la prima de cada cliente (ST_Within)</li> <li>• Encontrar el número de zonas con riesgo de inundación que están dentro de cada zona de oficinas (ST_Overlaps)</li> <li>• Encontrar la oficina más cercana respecto al lugar de residencia de un cliente determinado, suponiendo que la oficina esté situada en el centro geométrico de la zona de oficinas (ST_Distance)</li> <li>• Encontrar los clientes cuya ubicación está cerca del límite de una zona determinada con riesgo de inundación (ST_Buffer, ST_Intersects)</li> <li>• Encontrar los clientes expuestos a un riesgo alto que están dentro de una distancia especificada respecto a una oficina determinada (ST_Within)</li> </ul>
	<p>Todos estos pasos hacen uso del procedimiento almacenado sqlRunSpatialQueries.</p>
	<p>Estos pasos realizan el análisis espacial utilizando predicados y funciones espaciales del SQL de DB2. El optimizador de consultas de DB2 saca provecho del índice espacial definido sobre las columnas espaciales para mejorar el rendimiento de la consulta cuando sea posible.</p>
Exportar datos espaciales a archivos de forma	<ul style="list-style-type: none"> <li>• Exportar la vista HIGHRISKCUSTOMERS a archivos de forma</li> </ul> <p>Este paso muestra un ejemplo de exportación de la vista HIGHRISKCUSTOMERS a archivos de forma. Exportar datos desde un formato de base de datos a otro formato de archivo permite que la información pueda ser utilizada por otras herramientas (como, por ejemplo, ArcExplorer para DB2).</p> <p>Este paso forma parte del programa runGseDemo.c, pero está inhabilitado con fines de consulta solamente. Puede modificar el programa de ejemplo para especificar la ubicación del archivo de formas de exportación y ejecutar de nuevo el programa de ejemplo.</p>



---

## Capítulo 15. Identificación de problemas de DB2 Spatial Extender

Si tiene un problema al trabajar con DB2 Spatial Extender, necesita determinar la causa del problema.

Puede determinar la causa de los problemas de DB2 Spatial Extender de estas maneras:

- Puede utilizar la información de los mensajes para diagnosticar el problema.
- Cuando se utilizan procedimientos almacenados y funciones de Spatial Extender, DB2 proporciona información sobre la ejecución satisfactoria o errónea del procedimiento almacenado o función. La información devuelta consta de un código de mensaje (en forma de número entero), el texto del mensaje, o ambas cosas, según la interfaz que utilice para trabajar con DB2 Spatial Extender.
- Puede examinar el archivo de notificación de administración de DB2, en el cual se registra información de diagnóstico sobre errores.
- Si tiene un problema recurrente y reproducible de Spatial Extender, el soporte técnico de IBM le puede solicitar que utilice el programa de rastreo de DB2 para facilitar el diagnóstico del problema.

---

### Cómo interpretar mensajes de DB2 Spatial Extender

Puede trabajar con DB2<sup>®</sup> Spatial Extender a través de cuatro interfaces diferentes:

- Procedimientos almacenados de DB2 Spatial Extender
- Funciones de DB2 Spatial Extender
- Procesador de línea de mandatos (Command Line Processor, CLP) de DB2 Spatial Extender
- Centro de control de DB2

Todas las interfaces emiten mensajes de DB2 Spatial Extender para ayudar al usuario a determinar si la operación espacial solicitada se ejecutó satisfactoriamente o produjo un error.

La tabla siguiente describe cada parte del texto de este mensaje de ejemplo de DB2 Spatial Extender:

GSE0000I: La operación se ejecutó satisfactoriamente.

*Tabla 9. Partes del texto del mensaje de DB2 Spatial Extender*

Parte del mensaje	Descripción
GSE	Identificador del mensaje. Todos los mensajes de DB2 Spatial Extender comienzan con el prefijo de tres letras GSE.
0000	Número del mensaje. Es un número de cuatro dígitos comprendido entre 0000 y 9999.

Tabla 9. Partes del texto del mensaje de DB2 Spatial Extender (continuación)

Parte del mensaje	Descripción
I	Tipo de mensaje. Es una letra individual que indica la gravedad del mensaje:
C	Mensajes de error críticos
N	Mensajes de error no críticos
W	Mensajes de aviso
I	Mensajes informativos
La operación se ejecutó satisfactoriamente.	Explicación del mensaje.

Es una breve descripción que aparece en el texto del mensaje. El usuario puede obtener más información sobre el mensaje, que incluye una explicación detallada y sugerencias para evitar o corregir el problema. Para visualizar esa información:

1. Abra un indicador de mandatos del sistema operativo.
2. Escriba el mandato de ayuda de DB2 junto con el identificador y número del mensaje para visualizar más información sobre el mensaje. Por ejemplo:

```
DB2 "? GSEnnnn"
```

donde *nnnn* es el número de mensaje.

Puede escribir en mayúsculas o minúsculas el identificador del mensaje GSE y la letra que indica el tipo de mensaje. Si escribe DB2 "? GSE0000I" produce el mismo resultado que escribir db2 "? gse0000i".

Puede omitir la letra que sigue al número de mensaje cuando escriba el mandato. Por ejemplo, si escribe DB2 "? GSE0000" obtendrá el mismo resultado que si escribe DB2 "? GSE0000I".

Supongo que el código de mensaje es GSE4107N. Cuando escribe DB2 "? GSE4107N" en el indicador de mandatos, se muestra la información siguiente:

```
GSE4107N El valor de tamaño de cuadrícula "<tamaño-cuadrícula>" no es válido en el lugar que se ha utilizado.
```

Explicación: el tamaño de retícula especificado "<tamaño-retícula>" no es válido.

Se realizó una de las siguientes especificaciones no válidas cuando se creó el índice reticular con la sentencia CREATE INDEX:

- Se ha especificado un número menor que 0 (cero) como tamaño de retícula para el primer, segundo o tercer nivel reticular.
- Se ha especificado 0 (cero) como tamaño de retícula para el primer nivel de retícula.
- El tamaño de retícula especificado para el segundo nivel reticular es menor que el tamaño de retícula del primer nivel, pero no es 0 (cero).
- El tamaño de retícula especificado para el tercer nivel reticular es menor que el tamaño de retícula del segundo nivel, pero no es 0 (cero).
- El tamaño de retícula especificado para el tercer nivel reticular es mayor que 0 (cero) pero el tamaño de retícula especificado para el

segundo nivel reticular es 0 (cero).

Respuesta del usuario: especifique un valor válido para el tamaño de retícula.

msgcode: -4107

sqlstate: 38SC7

Si la información es demasiado larga para visualizarse en una sola pantalla y su sistema operativo permite utilizar el mandato **more** y redireccionamientos de mandato, escriba:

```
db2 "? GSEnnn" | more
```

La especificación **more** hace que la visualización haga una pausa después de cada pantalla de datos para que el usuario pueda leer la información.

---

## Parámetros de salida de procedimientos almacenados de DB2 Spatial Extender

Los procedimientos almacenados de DB2<sup>®</sup> Spatial Extender se invocan implícitamente cuando se habilita y se utiliza Spatial Extender desde el Centro de control de DB2 o cuando se utiliza el CLP de DB2 Spatial Extender (db2se). Puede invocar procedimientos almacenados explícitamente en un programa de aplicación o desde la línea de mandatos de DB2.

Este tema describe cómo diagnosticar problemas cuando se invocan explícitamente procedimientos almacenados desde programas de aplicación o desde la línea de mandatos de DB2. Para diagnosticar procedimientos almacenados que se han invocado implícitamente, utilice los mensajes que devuelve el CLP de DB2 Spatial Extender o los mensajes que devuelve el Centro de control de DB2. Estos mensajes se tratan en temas separados.

Los procedimientos almacenados de DB2 Spatial Extender tienen dos parámetros de salida: el código del mensaje (msg\_code) y el texto del mensaje (msg\_text). Los valores de los parámetros indican si el procedimiento almacenado se ha ejecutado satisfactoriamente o no.

### **código\_mje**

El parámetro código\_mje es un valor entero que puede ser positivo, negativo o cero (0). Los valores positivos se utilizan para los avisos, los valores negativos se utilizan para los errores (críticos o no) y el cero (0) se utiliza para los mensajes informativos.

El valor absoluto de código\_mje se incluye en texto\_mje y constituye el número del mensaje. Por ejemplo

- Si código\_mje es 0, el número de mensaje es 0000.
- Si código\_mje es -219, el número de mensaje es 0219. El valor negativo de código\_mje indica que el mensaje indica un error, crítico o no.
- Si código\_mje es +1036, el número de mensaje es 1036. El valor positivo de código\_mje indica que el mensaje es un aviso.

Los valores de los códigos de mensaje emitidos por los procedimientos almacenados de Spatial Extender se clasifican en tres categorías, tal como se muestra en la tabla siguiente:

Tabla 10. Códigos de mensaje de procedimiento almacenado

Códigos	Categoría
0000 – 0999	Mensajes comunes
1000 – 1999	Mensajes administrativos
2000 – 2999	Mensajes de importación y exportación

#### texto\_mje

El parámetro texto\_mje comprende el identificador del mensaje, el número de mensaje, el tipo de mensaje y la explicación. El ejemplo siguiente muestra un valor de texto\_mje de un procedimiento almacenado:

```
GSE0219N  Una sentencia EXECUTE IMMEDIATE
          ha fallado.  SQLERROR
= "<error-sql>".
```

En el parámetro texto\_mje se incluye una breve explicación. El usuario puede obtener más información sobre el mensaje, que incluye una explicación detallada y sugerencias para evitar o corregir el problema.

Para conocer detalles sobre las partes del parámetro msg\_text y obtener información sobre cómo recuperar más información sobre el mensaje, consulte el apartado "Cómo interpretar los mensajes de DB2 Spatial Extender".

## Utilización de procedimientos almacenados en aplicaciones

Cuando el usuario invoca a un procedimiento almacenado de DB2 Spatial Extender desde una aplicación, recibe msg\_code y msg\_text como parámetros de salida. El usuario puede:

- Programar la aplicación para devolver los valores de parámetros de salida al usuario de la aplicación.
- Empezar acciones de acuerdo con el tipo de valor de código de mensaje devuelto.

## Utilización de procedimientos almacenados desde la línea de mandatos de DB2

Cuando el usuario invoca un procedimiento almacenado de DB2 Spatial Extender desde la línea de mandatos de DB2, recibe los parámetros de salida código\_mje y texto\_mje. Estos parámetros indican si el procedimiento almacenado se ha ejecutado satisfactoriamente o no.

Suponga que se conecta a una base de datos y desea invocar al procedimiento almacenado ST\_disable\_db. El ejemplo siguiente utiliza un mandato CALL de DB2 para inhabilitar la base de datos para operaciones espaciales y muestra los resultados obtenidos. Se utiliza el parámetro force con un valor de 0, junto con dos signos de interrogación al final del mandato CALL para representar los parámetros de salida código\_mje y texto\_mje. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

```
call db2gse.st_disable_db(0, ?, ?)
```

Valor de los parámetros de salida

-----

Nombre de parámetro : MSGCODE

Valor de parámetro : 0

Nombre de parámetro : MSGTEXT  
Valor de parámetro : GSE0000I La operación se ejecutó satisfactoriamente.

Estado de retorno = 0

Suponga que el valor devuelto por texto\_mje es GSE2110N. Utilice el mandato help de DB2 para visualizar más información sobre el mensaje. Por ejemplo:

"? GSE2110"

Se visualiza la información siguiente:

GSE2110N El sistema de referencia espacial para la  
la geometría no es válido en la fila "<numero-fila>".  
El identificador numérico del sistema de referencia espacial  
es "<id-srs>".

Explicación: En la fila *numero-fila*, la geometría que se debe exportar utiliza un sistema de referencia espacial no válido. La geometría no se puede exportar.

Respuesta del usuario: Corrija la geometría indicada o excluya la fila de la operación de exportación modificando la sentencia SELECT según proceda.

msg\_code: -2110

sqlstate: 38S9A

---

## Mensajes de las funciones de DB2 Spatial Extender

Los mensajes emitidos por las funciones de DB2<sup>®</sup> Spatial Extender suelen estar incluidos en un mensaje de SQL. El código de SQL (SQLCODE) devuelto en el mensaje indica se ha producido un error o aviso para la función. Por ejemplo:

- El SQLCODE -443 (número de mensaje SQL0443) indica que se ha producido un error con la función.
- El SQLCODE +462 (número de mensaje SQL0462) indica que existe una condición de aviso asociada a la función.

La tabla siguiente explica las partes significativas de este mensaje de ejemplo:

DB21034E El mandato se ha procesado como una sentencia SQL porque no era un mandato válido para el Procesador de línea de mandatos. Durante el proceso SQL se ha devuelto: SQL0443N Rutina "DB2GSE.GSEGEOMFROMWKT" (nombre específico "GSEGEOMWKT1") ha devuelto un error SQLSTATE con el texto de diagnóstico "GSE3421N El polígono no está cerrado.". SQLSTATE=38SSL

Tabla 11. Partes significativas de los mensajes de las funciones de DB2 Spatial Extender

Parte del mensaje	Descripción
SQL0443N	El SQLCODE indica el tipo de problema.
GSE3421N	Es el número de mensaje y tipo de mensaje de DB2 Spatial Extender.  Los números de mensaje de las funciones están comprendidos entre GSE3000 y GSE3999. Además, se pueden emitir mensajes comunes al trabajar con funciones de DB2 Spatial Extender. Los números de mensaje de los mensajes comunes están comprendidos entre GSE0001 y GSE0999.
Polígono no cerrado	Es la explicación del mensaje de DB2 Spatial Extender.

Tabla 11. Partes significativas de los mensajes de las funciones de DB2 Spatial Extender (continuación)

Parte del mensaje	Descripción
SQLSTATE=38SSL	<p>Código SQLSTATE que identifica con más detalle el error. Se emite un código SQLSTATE para cada sentencia o fila.</p> <ul style="list-style-type: none"> <li>• Los códigos SQLSTATE para errores de funciones de Spatial Extender tienen la forma 38Sxx, donde x es una letra o un número.</li> <li>• Los códigos SQLSTATE para avisos de funciones de Spatial Extender tienen la forma 01HSx, donde x es una letra o un número.</li> </ul>

## Ejemplo de mensaje de error SQL0443

Suponga que intenta insertar los valores de un polígono en la tabla POLYGON\_TABLE, tal como se muestra a continuación:

```
INSERT INTO polygon_table ( geometry )
VALUES ( ST_Polygon ( 'polygon (( 0 0, 0 2, 2 2, 1 2)) ' ) )
```

Esto origina un mensaje de error debido a que el usuario no ha proporcionado el valor final para cerrar el polígono. El mensaje de error devuelto es:

```
DB21034E El mandato se ha procesado como una sentencia SQL porque
no era un mandato válido para el Procesador de línea de mandatos.
Durante el proceso SQL
se ha devuelto: SQL0443N Rutina "DB2GSE.GSEGEOMFROMWKT"
(nombre específico "GSEGEOMWKT1") ha devuelto un error
SQLSTATE con el texto de diagnóstico "GSE3421N El polígono no está cerrado.".
SQLSTATE=38SSL
```

El número de mensaje de SQL SQL0443N indica que se ha producido un error y el mensaje incluye el texto del mensaje de Spatial Extender GSE3421N El polígono no está cerrado.

Si recibe este tipo de mensaje:

1. Localice el número de mensaje GSE dentro del mensaje de error de DB2 o SQL.
2. Utilice el mandato de ayuda de DB2 (DB2 ?) para ver la explicación del mensaje de Spatial Extender y la respuesta del usuario. Utilizando el ejemplo anterior, escriba el mandato siguiente en el indicador de línea de mandatos del sistema operativo:

```
DB2 "? GSE3421"
```

El mensaje se repite, junto con una explicación detallada y la respuesta del usuario recomendada.

---

## Mensajes del CLP de DB2 Spatial Extender

El CLP de DB2<sup>®</sup> Spatial Extender (db2se) devuelve mensajes para:

- Procedimientos almacenados, si se invocan implícitamente.
- Información sobre formas, si ha invocado al programa de submandato **shape\_info** desde el CLP de DB2 Spatial Extender. Estos mensajes son informativos.
- Operaciones de migración.
- Operaciones de importación y exportación de formas con el cliente.



## Ejemplos de mensajes de procedimiento almacenado emitidos por el CLP de DB2 Spatial Extender

La mayoría de los mensajes que se devuelven desde el CLP de DB2 Spatial Extender son para procedimientos almacenados de DB2 Spatial Extender. Cuando invoca a un procedimiento almacenado desde el CLP de DB2 Spatial Extender, recibe un texto de mensaje que indica el éxito o fracaso del procedimiento almacenado.

El texto del mensaje comprende el identificador del mensaje, el número de mensaje, el tipo de mensaje y la explicación. Por ejemplo, si habilita una base de datos utilizando el mandato `db2se enable_db testdb`, el texto del mensaje devuelto por el CLP de Spatial Extender es:

Se está habilitando la base de datos. Espere, por favor ...

```
GSE1036W  La operación se ha realizado satisfactoriamente. Sin embargo, los valores de determinados parámetros del gestor de base de datos y de configuración de base de datos deberían aumentarse.
```

Del mismo modo, si habilita una base de datos utilizando el mandato `db2se disable_db testdb`, el texto del mensaje devuelto por el CLP de Spatial Extender es:

```
GSE0000I La operación se ejecutó correctamente.
```

Es una breve descripción que aparece en el texto del mensaje. El usuario puede obtener más información sobre el mensaje, que incluye una explicación detallada y sugerencias para evitar o corregir el problema. Los pasos para recuperar esta información, e información detallada sobre cómo interpretar las partes del texto del mensaje, se tratan en un tema separado.

Si está invocando a procedimientos almacenados desde un programa de aplicación o desde la línea de mandatos de DB2, en un tema separado se trata el diagnóstico de los parámetros de salida.

## Ejemplo de mensajes de información sobre formas emitidos por el CLP del Spatial Extender

Suponga que desea visualizar información sobre un archivo de formas llamado `office`. Desde el CLP de Spatial Extender (`db2se`) emitiría este mandato:

```
db2se shape_info -nombreArchivo /tmp/offices
```

Esto es un ejemplo de la información que se visualiza:

Información del archivo de formas

```
-----  
Código de archivo = 9994  
Longitud del archivo (palabras de 16 bits) = 484  
Versión del archivo de formas = 1000  
Tipo de forma = 1 (ST_POINT)  
Número de registros = 31
```

```
Coordenada X mínima = -87.053834  
Coordenada X máxima = -83.408752  
Coordenada Y mínima = 36.939628  
Coordenada Y máxima = 39.016477  
Las formas no tienen coordenadas Z.  
Las formas no tienen coordenadas M.
```

Aparece el archivo de índices de formas (extensión `.shx`).

#### Información del archivo de atributos

-----  
Código del archivo dBase = 3  
Fecha de la última actualización = 1901-08-15  
Número de registros = 31  
Número de bytes en la cabecera = 129  
Número de bytes en cada registro = 39  
Número de columnas = 3

Núm. col.	Nombre col.	Tipo de dato	Long.	Decimal
1	NAME	C ( Character)	16	0
2	EMPLOYEES	N ( Numeric)	11	0
3	ID	N ( Numeric)	11	0

Definición del sistema de coordenadas: "GEOGCS["GCS\_North\_American\_1983",  
DATUM["D\_North\_American\_1983",SPHEROID["GRS\_1980",6378137,298.257222101]],  
PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]]"

## Ejemplos de mensajes de actualización emitidos por el CLP de Spatial Extender

Cuando el usuario invoca a mandatos que realizan operaciones de migración, se emiten mensajes que indican el éxito o el fracaso de esa operación.

Supongamos que la base de datos espacial *mydb* utiliza el mandato siguiente:

```
db2se upgrade mydb -messagesFile /tmp/db2se_upgrade.msg
```

El texto del mensaje devuelto por el CLP de Spatial Extender es:

```
Actualizando la base de datos. Espere, por favor ...  
GSE0000I La operación se ejecutó correctamente.
```

---

## Mensajes del Centro de control de DB2

### Mensajes de DB2 Spatial Extender

Cuando el usuario trabaja con DB2<sup>®</sup> Spatial Extender mediante el Centro de control de DB2, los mensajes aparecen en la ventana Mensaje de DB2. La mayoría de los mensajes que recibe el usuario son mensajes de DB2 Spatial Extender. Ocasionalmente, el usuario recibe un mensaje de SQL. Se emiten mensajes de SQL cuando se produce un error referente al uso de licencias, bloqueos, o cuando un servicio DAS no está disponible. Las secciones siguientes proporcionan ejemplos de cómo aparecen los mensajes de DB2 Spatial Extender y de SQL en el Centro de control de DB2.

Cuando recibe un mensaje de DB2 Spatial Extender a través del Centro de control, el texto completo del mensaje aparece en el área de texto de la ventana Mensaje de DB2, por ejemplo:

```
GSE0219N Una sentencia EXECUTE IMMEDIATE  
ha fallado. SQLERROR  
= "<error-sql>".
```

### Mensajes de SQL

Cuando recibe un mensaje de SQL a través del Centro de control que pertenece a DB2 Spatial Extender:

- El identificador, número y tipo del mensaje aparecen en el lado izquierdo de la ventana Mensaje de DB2, por ejemplo: SQL0612N.
- El texto del mensaje aparece en el área de texto de la ventana Mensaje de DB2.

El texto del mensaje que aparece en la ventana Mensaje de DB2 puede incluir el texto de mensaje de SQL y el SQLSTATE, o bien puede contener el texto del mensaje, la explicación detallada y la respuesta del usuario.

Esto es un ejemplo de un mensaje de SQL que contiene el texto del mensaje de SQL y el SQLSTATE:

```
[IBM][Controlador CLI][DB2/NT] SQL0612N "<nombre>" es un nombre duplicado.
SQLSTATE=42711
```

Esto es un ejemplo de un mensaje de SQL que contiene el texto del mensaje, la explicación detallada y la respuesta del usuario:

```
SQL8008N
```

El producto "DB2 Spatial Extender" no tiene instalada una clave de licencia válida y el periodo de evaluación ha vencido.

Explicación:

No se ha podido encontrar una clave de licencia válida y el periodo de evaluación ha vencido.

Respuesta del usuario:

Instale una clave de licencia para la versión del producto totalmente autorizada. Puede obtener una clave de licencia para el producto poniéndose en contacto con el representante de IBM® o con el distribuidor autorizado.

---

## Rastreo de problemas de DB2 Spatial Extender con el mandato db2trc

Si se produce un problema repetitivo y reproducible de DB2 Spatial Extender, puede utilizar el recurso de rastreo de DB2 para recoger información sobre el problema.

El recurso de rastreo de DB2 se activa con el mandato del sistema **db2trc**. El recurso de rastreo de DB2 puede:

- Rastrear sucesos
- Volcar los datos de rastreo en un archivo
- Formatear los datos de rastreo para convertirlos en un formato legible.

### Restricción:

- Active este recurso solo cuando se lo indique el servicio técnico de DB2.
- En los sistemas operativos UNIX, debe tener autorización SYSADM, SYSCTRL o SYSMANT para rastrear una instancia de DB2.
- En los sistemas operativos Windows no es necesaria ninguna autorización especial.

Para realizar esta tarea:

1. Cierre todas las demás aplicaciones.
2. Active el rastreo.

El servicio técnico de DB2 le proporcionará los parámetros específicos para este paso. El mandato básico es:

```
db2trc on
```

**Restricción:** El mandato **db2trc** debe especificarse en un indicador de mandatos del sistema operativo o en un script de shell. No se puede utilizar en la interfaz de línea de mandatos de DB2 Spatial Extender (db2se) ni en el CLP de DB2.

Puede volcar datos de rastreo en la memoria o en un archivo. El método preferido es volcar datos de rastreo en la memoria. Si el programa que se está reproduciendo suspende la actividad de la estación de trabajo y le impide volcar el rastreo, realice el rastreo en un archivo.

3. Reproduzca el problema.
4. Vuelva el rastreo en un archivo inmediatamente después de que se produzca el problema.

Por ejemplo:

```
db2trc dump january23trace.dmp
```

Este mandato crea un archivo (*january23trace.dmp*) en el directorio actual con el nombre especificado por el usuario y vuelca la información de rastreo en ese archivo. Puede especificar un directorio diferente indicando la vía de acceso del archivo. Por ejemplo, para colocar el archivo de rastreo en el directorio */tmp/spatial/errors*, la sintaxis es:

```
db2trc dump /tmp/spatial/errors/january23trace.dmp
```

5. Desactive el rastreo.

Por ejemplo:

```
db2trc off
```

6. Formatee los datos como un archivo ASCII. Puede clasificar los datos de dos maneras:

- Utilice la opción *flw* para clasificar los datos según el proceso o hebra. Por ejemplo:

```
db2trc flw january23trace.dmp january23trace.flw
```

- Utilice la opción *fmt* para listar los sucesos cronológicamente. Por ejemplo:

```
db2trc fmt january23trace.dmp january23trace.fmt
```

---

## El archivo de notificaciones de administración

La información de diagnóstico acerca de errores se graba en el archivo de notificaciones de administración. Esta información se utiliza para la determinación de problemas y está pensada para el soporte técnico de DB2®.

El archivo de notificaciones de administración contiene información de texto registrada por DB2 así como DB2 Spatial Extender. Está ubicado en el directorio especificado por el parámetro de configuración del gestor de la base de datos *DIAGPATH*. En los sistemas Windows® NT, Windows 2000 y Windows XP, el archivo de notificación de administración de DB2 se encuentra en la anotación cronológica de sucesos y se puede revisar con el Visor de sucesos de Windows.

La información que DB2 registra en el archivo de anotaciones cronológicas de administración está determinada por los valores *DIAGLEVEL* y *NOTIFYLEVEL*.

Utilice un editor de texto para visualizar el archivo en la máquina donde sospecha que se ha producido un problema. Los sucesos más recientes aparecen registrados al final del archivo. Generalmente, cada entrada del archivo tiene estas partes:

- Indicación de fecha y hora.

- La ubicación que notifica el error. Los identificadores de aplicación permiten al usuario emparejar entradas pertenecientes a una aplicación en los archivos de anotaciones cronológicas de servidores y clientes.
- Un mensaje de diagnóstico (que generalmente empieza por "DIA" o "ADM") que describe el error.
- Datos auxiliares disponibles, tales como estructuras de datos SQLCA y punteros que indican la posición de archivos de vuelco o de captura adicionales.

Si el comportamiento de la base de datos es normal, este tipo de información no es importante y se puede pasar por alto.

El archivo de notificaciones de administración crece constantemente. Cuando su tamaño sea demasiado grande, haga una copia del archivo y luego borre el archivo. El sistema generará automáticamente un nuevo archivo la próxima vez que sea necesario.



---

## Capítulo 16. DB2 Geodetic Data Management Feature

Este capítulo proporciona información preliminar sobre DB2 Geodetic Data Management Feature; describe su función, cuándo se utiliza y explica conceptos geodésicos.

---

### DB2 Geodetic Data Management Feature

DB2® Geodetic Data Management Feature permite tratar la tierra como si fuera un globo. Utilizando los mismos tipos de datos y funciones espaciales que para el resto de operaciones Spatial Extender, puede utilizar Geodetic Data Management Feature para ejecutar consultas transparentes de datos transpolares y de datos que crucen el meridiano 180. Puede realizar el mantenimiento de datos que hagan referencia a una ubicación precisa sobre la superficie de la tierra.

Geodetic Data Management Feature debe su nombre a la disciplina denominada *geodesia*. La geodesia es el estudio del tamaño y de la forma de la Tierra (o de cualquier cuerpo modelado por un elipsoide, como el sol o una esfera celeste). Geodetic Data Management Feature está diseñada para manejar objetos definidos en la superficie de la tierra con un alto grado de precisión.

Para lograr esta precisión, Geodetic Data Management Feature utiliza un sistema de coordenadas de latitud y longitud en un modelo de tierra elipsoidal o *sistema de referencia geodésico*, en lugar de un sistema de coordenadas *x- e y-* plano. Un modelo elipsoidal evita distorsiones, inexactitudes e imprecisiones que se pueden introducir con las proyecciones planas.

Para acceder a operaciones geodésicas en vez de espaciales, deberá definir un sistema de referencia espacial geodésico para sus datos. Estos sistemas tienen ID de sistema de referencia espacial (SRID) comprendidos entre 2000000000 y 2000001000. DB2 Geodetic Data Management Feature proporciona 318 sistemas de referencia espacial geodésicos predefinidos.

DB2 Spatial Extender debe estar instalado para poder utilizar DB2 Geodetic Data Management Feature. Para habilitar Geodetic Data Management Feature, debe adquirir una licencia distinta.

---

### Cuándo se utiliza DB2 Geodetic Data Management Feature y cuándo se utiliza DB2 Spatial Extender

DB2® Spatial Extender y DB2 Geodetic Data Management Feature gestionan datos de sistemas de información geográfica (GIS) en una base de datos de DB2. Cada extensor utiliza distintas tecnologías básicas para resolver diferentes problemas y se complementan mutuamente:

- Geodetic Data Management Feature considera que la tierra es un globo. Utiliza un sistema de coordenadas de latitud y longitud en un modelo elipsoidal de la tierra. Las operaciones geométricas son precisas, independientemente de la ubicación. Se basa en la biblioteca Hipparchus, cuya licencia es propiedad de Geodyssey Limited. Para obtener más información, consulte <http://www.geodyssey.com>.

Geodetic Data Management Feature se usa principalmente con aplicaciones y conjuntos de datos globales que cubren áreas grandes de la tierra, en los que una simple proyección de mapa no puede proporcionar la precisión que requiere la aplicación.

- Spatial Extender considera que la tierra es un mapa plano. Utiliza la geometría planimétrica (de nivel plano), es decir, que se aproxima a la superficie redonda de la tierra proyectándola en un nivel plano. Esta proyección produce distorsiones, que pueden variar en función de la extensión de los datos, aunque las distorsiones aumentan generalmente hacia los bordes de la región proyectada. Todas las proyecciones de nivel plano sufren algún tipo de distorsión. Spatial Extender se basa en la biblioteca de formas ESRI, cuya licencia es propiedad de ESRI. Para obtener más información espacial, consulte <http://www.esri.com>.

Spatial Extender se utiliza principalmente para conjuntos de datos regionales y locales que se representan correctamente en coordenadas proyectadas y para aplicaciones en las que la precisión de la ubicación no reviste importancia. Por ejemplo, cuando una compañía de seguros médicos desea conocer las ubicaciones de los hospitales y las clínicas de una región o provincia determinada.

---

## Sistemas de referencia geodésicos

Un sistema de referencia geodésico es un sistema de referencia que describe la superficie de la tierra. Durante siglos se han desarrollado muchos sistemas de referencia de este tipo a medida que la ciencia ha ido desarrollando nuevas herramientas para medir la tierra. Se han utilizado mediciones terrestres y por satélite para crear sistemas de referencia, que a su vez se han destinado a crear proyecciones de nivel plano.

Los sistemas de referencia geodésicos se basan en una aproximación de la forma general de la tierra mediante un elipsoide de rotación (también denominada *esferoide*). Un esferoide es la forma tridimensional que se describe por una elipse cuando se ha girado alrededor de uno de sus ejes.

Todos los objetos espaciales que defina deben hacer alusión a un sistema de referencia específico. Debe especificar un sistema de referencia por el identificador de sistemas de referencia espacial (SRID). Puede que cualquier sistema de referencia que esté soportado por DB2<sup>®</sup> Geodetic Data Management Feature. Estos sistemas tienen SRID comprendidos entre 2000000000 y 2000001000.

- "Sistemas de referencia soportados por DB2 Geodetic Data Management Feature" lista los 318 sistemas de referencia espacial geodésicos predefinidos que proporciona Geodetic Data Management Feature.
- También puede definir un nuevo sistema de referencia creando un sistema de referencia espacial con un ID comprendido entre 2000000318 y 2000001000.

**Restricción:** Las funciones que utilizan más de un objeto geoespacial como argumentos no pueden manejar combinaciones de sistemas de referencia. Geodetic Data Management Feature no realiza conversiones de sistemas de referencia.



## Latitud y longitud geodésica

El sistema de referencia de coordenadas de DB2 Geodetic Data Management Feature utiliza la *latitud* y la *longitud* geodésicas para describir ubicaciones de la Tierra. La latitud y la longitud geodésicas se basan siempre en un sistema de referencia específico.

### Latitud geodésica

La latitud geodésica de un punto es el ángulo entre el plano ecuatorial y la línea perpendicular que cruza la línea normal en el punto de la superficie de la tierra.

### Longitud geodésica

La *longitud geodésica* es el ángulo del plano ecuatorial entre la línea *a* que conecta el centro de la tierra y el meridiano de origen, y la línea *b* que conecta el centro con el meridiano en el que reside el punto. Un *meridiano* es una ruta directa sobre la superficie del sistema de referencia que constituye la distancia más corta entre los polos.

El elipsoide de la Figura 17 muestra los ángulos que representan la latitud y la longitud geodésica. El ángulo de la latitud geodésica no empieza en el mismo centro debido a la forma elipsoidal de la tierra.

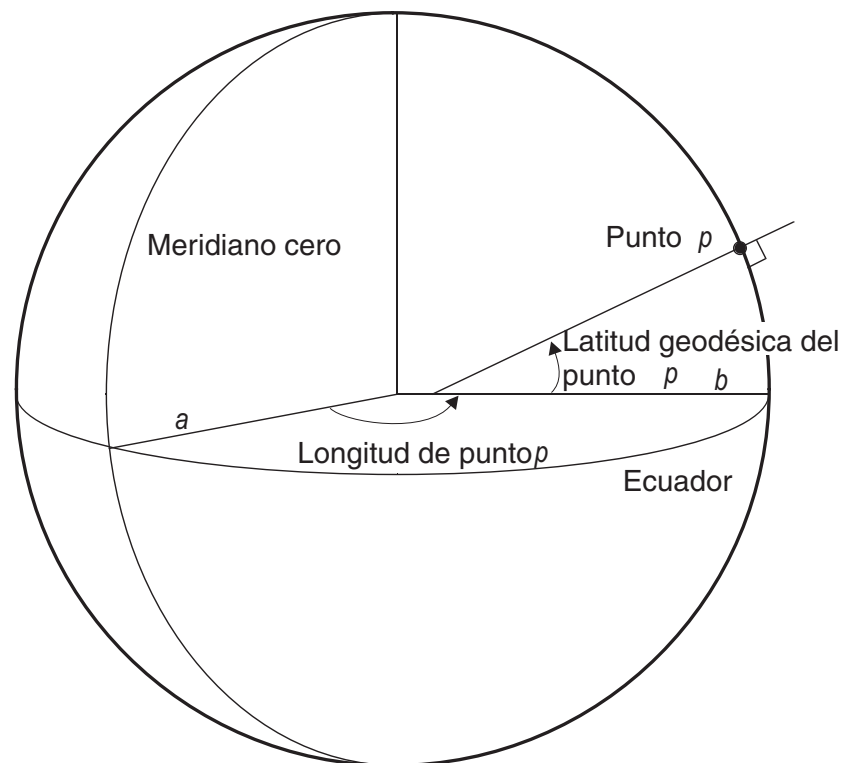


Figura 17. Ángulos de latitud y longitud geodésica

Las coordenadas de latitud y longitud se expresan en grados con una fracción decimal. Hay 360 grados de longitud, empezando por el meridiano de origen (longitud 0°) y siguiendo hacia el este en una dirección positiva hasta los 180° y hacia el oeste en valores negativos hasta -180°. Los grados de latitud empiezan en el ecuador (latitud 0°) y siguen hacia el polo norte (latitud 90°) y hacia el polo sur (latitud -90°).

---

## Distancias geodésicas

DB2<sup>®</sup> Geodetic Data Management Feature mide la distancia entre dos puntos a lo largo de un *geodésico*. Un geodésico es la trayectoria más corta entre dos puntos de la forma elipsoidal de la tierra; es posible que esta trayectoria más corta no siga una latitud constante aunque los dos puntos finales estén en la misma latitud.

Puesto que los segmentos lineales se computan como geodésicos, es posible que un polígono de cuatro puntos con los puntos ampliamente separados no circunde la región de interés, tal como muestra la Figura 18. Este polígono cubre una región con líneas de longitud que están separadas unos 120 grados; los dos puntos superiores tienen los mismos valores de latitud y los dos puntos inferiores también. La geodésica entre las dos líneas de longitud sigue la curva de la forma elipsoidal de la tierra. La latitud aumenta a lo largo de la geodésica hasta 20 grados más en el centro que en los extremos de la misma.

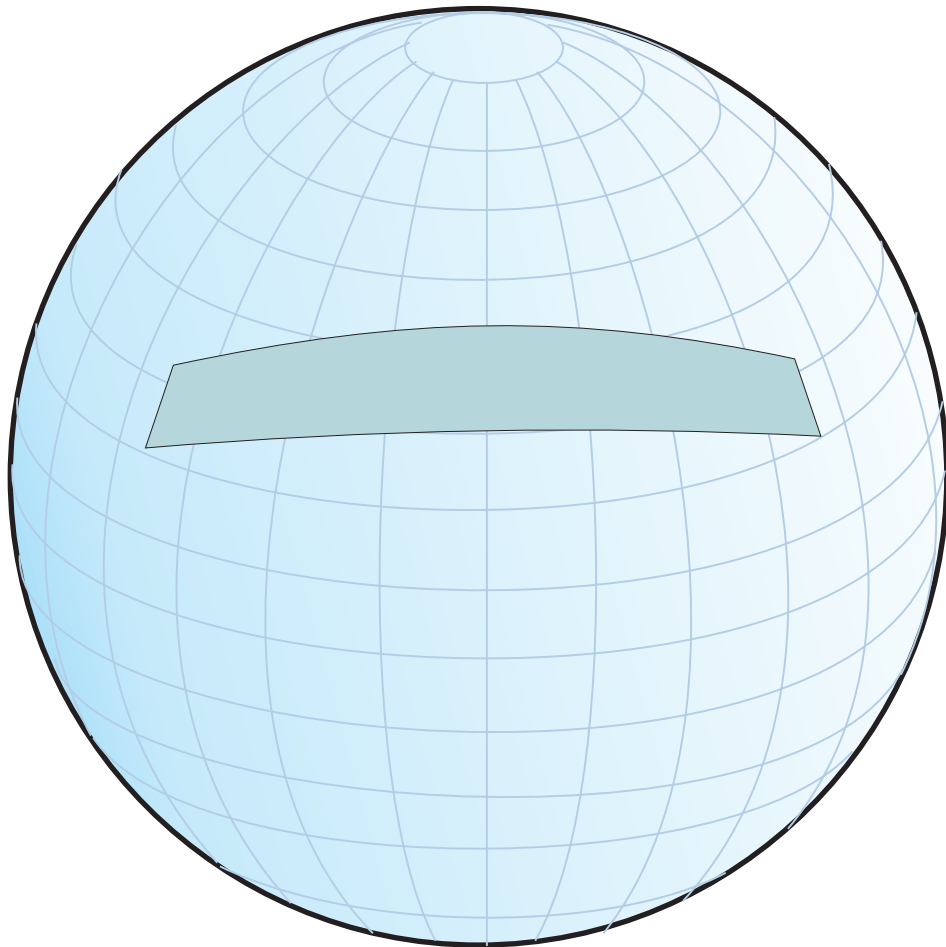


Figura 18. Región delimitada por un polígono con puntos ampliamente separados

Si desea representar una trayectoria que no sea geodésica, como por ejemplo, un segmento lineal que siga una latitud constante, deberá insertar puntos intermedios adicionales.

## Regiones geodésicas

Una región geodésica (polígono) es un área de la superficie de la tierra que tiene alguna característica específica para una aplicación. Ejemplos de regiones podrían ser un área de influencia comercial o el área avistada por un satélite durante un periodo de tiempo determinado.

Geodetic Data Management Feature define una región mediante una secuencia ordenada de puntos que forman un anillo cerrado. El orden en que se especifican los puntos de un polígono es significativo. Siguiendo un polígono de vértice a vértice en el orden definido, el área situada a la izquierda se encuentra dentro del polígono.

Puede utilizar un tipo de dato ST\_Polygon para definir una región delimitada por uno o varios anillos, tal como se muestra en la Figura 19. Defina el polígono mediante las coordenadas de latitud y longitud de los puntos (vértices) que forman sus anillos.

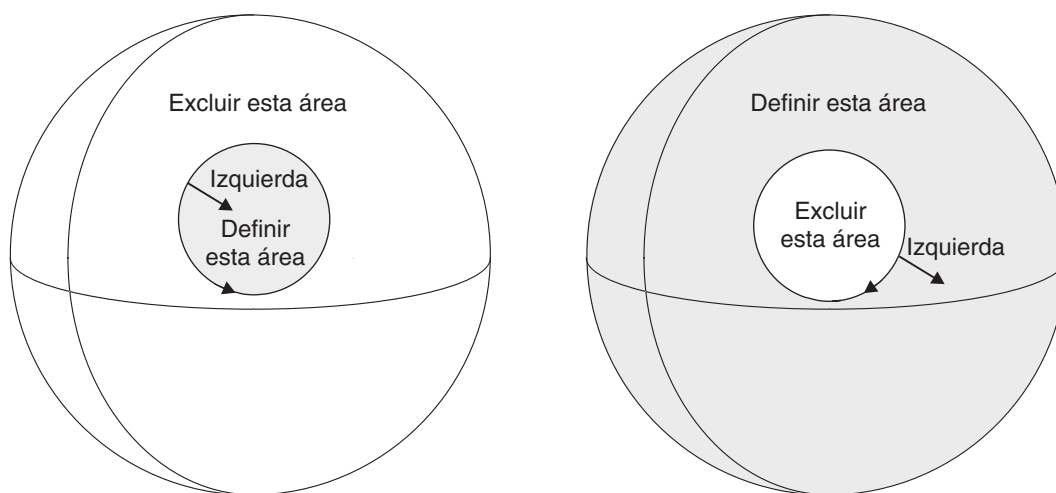


Figura 19. Áreas de definición y de exclusión

Un anillo divide la superficie de la tierra en dos regiones: una región dentro del polígono y otra fuera del polígono. La parte izquierda de la Figura 19 muestra un anillo con vértices especificados en orden contrario al de las agujas del reloj, de modo que todos los puntos de la izquierda están dentro del anillo. La parte derecha de la figura muestra un anillo cuyos vértices están especificados en orden contrario al de las agujas del reloj, de modo que todos los puntos de la izquierda están fuera del anillo.

Para definir una región como un polígono, debe especificar el orden de los vértices de cada anillo de modo que el interior del polígono esté a su izquierda cuando atravesase el anillo. Para definir una región excluida, debe especificar los vértices del anillo en orden inverso, tal como muestra la Figura 20 en la página 138. El interior del polígono siempre se encuentra a la izquierda. La Figura 20 en la página 138 muestra dos anillos, uno dentro del otro. El anillo más grande define el perímetro exterior del polígono y se ha dibujado en sentido contrario al de las agujas del reloj. El anillo más pequeño define el perímetro interno y se ha dibujado en el sentido de las agujas del reloj.

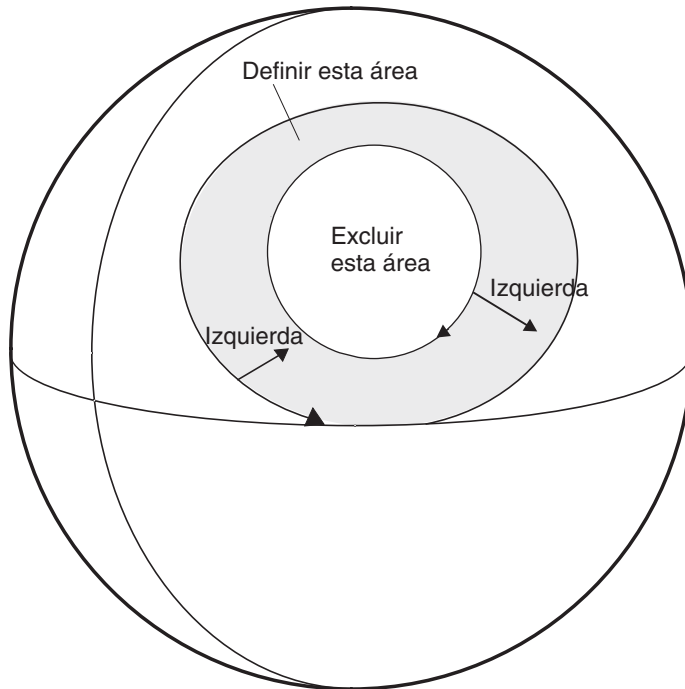


Figura 20. Definición de un área con varios anillos

Si crea un polígono más grande que un hemisferio, recibirá el siguiente mensaje de aviso. Es posible que desee conservar este polígono grande, pero el aviso aparece por si ha especificado sin advertirlo un orden equivocado en los vértices y en vez de un polígono pequeño ha generado un polígono grande.

GSE3733W "El polígono cubre más de la mitad de la tierra. Verifique la orientación en sentido contrario a las agujas del reloj de los puntos del vértice."

---

## Capítulo 17. Configuración de DB2 Geodetic Data Management Feature

Este capítulo ofrece instrucciones para configurar DB2 Geodetic Data Management Feature, migrar desde Informix Geodetic DataBlade y llenar columnas espaciales con datos geodésicos.

---

### Configuración y habilitación de DB2 Geodetic Data Management Feature

Para poder habilitar DB2 Geodetic Data Management Feature, debe:

- Instalar y configurar DB2 Versión 9.5.  
Debe instalar la base de datos DB2 en el sistema *antes* de instalar DB2 Spatial Extender y DB2 Geodetic Data Management Feature. Si piensa utilizar el Centro de control de DB2, cree y configure el Servidor de administración de DB2 (DAS). Para más información sobre cómo crear y configurar DATS, consulte la publicación *IBM DB2 Administration Guide: Implementation*
- Instalar y configurar DB2 Spatial Extender.  
DB2 Geodetic Data Management Feature está integrado en el mismo código de biblioteca que DB2 Spatial Extender. Por lo tanto, el CD de instalación de Spatial Extender incluye Geodetic Data Management Feature. Los requisitos de espacio de disco de Spatial Extender incluyen Geodetic Data Management Feature. Sin embargo, no puede utilizar Geodetic Data Management Feature hasta que adquiera y habilite una licencia de Geodetic Data Management Feature.
- Adquirir una licencia de DB2 Geodetic Data Management Feature.  
Cuando adquiera una licencia de DB2 Geodetic Data Management Feature, podrá habilitar la clave de licencia de Geodetic. Póngase en contacto con su representante de ventas si desea adquirir DB2 Geodetic Data Management Feature.

#### Restricción:

- DB2 Geodetic Data Management Feature sólo tiene licencia para DB2 Versión 9.5 Enterprise Server Edition.
- DB2 Geodetic Data Management Feature considera que la tierra es un globo, mientras que Spatial Extender considera que la superficie redonda de la tierra es un mapa plano. Si instala Geodetic Data Management Feature, podrá analizar datos espaciales con mayor precisión que un mapa plano.
- Un sistema DB2 Geodetic Data Management Feature consta del sistema de bases de datos DB2, DB2 Spatial Extender, DB2 Geodetic Data Management Feature, y, en la mayoría de las aplicaciones, un geonavegador.

Para obtener información adicional o modificada sobre la habilitación de DB2 Geodetic Data Management Feature, consulte el manual *Notas del release de DB2*.

Tras habilitar la licencia de DB2 Geodetic Data Management Feature, deberá llenar las columnas espaciales con datos geodésicos.

Para realizar esta tarea:

Habilite la licencia de DB2 Geodetic Data Management Feature de uno de los modos siguientes:

- Utilice el Centro de licencias del Centro de control de DB2. Consulte la ayuda en línea del Centro de licencias de DB2 para obtener más información sobre cómo habilitar la licencia de Geodetic.
- Ejecute el mandato **db2licm**.

## Migración de Informix Geodetic DataBlade a DB2 Geodetic Data Management Feature

### Requisitos previos

Debe transferir las aplicaciones DataBlade de Geodetic para utilizar los tipos de datos y funciones de DB2 Geodetic Data Management Feature.

### Restricciones

Si actualmente utiliza Informix Geodetic DataBlade, probablemente podrá migrar a DB2 Geodetic Data Management Feature si cumple los requisitos siguientes:

- Utiliza sólo tipos de datos GeoPoint, GeoLineseg, GeoString, GeoRing y GeoPolygon.
- Utiliza únicamente las funciones de Geodetic DataBlade que tienen un equivalente exacto o aproximado en DB2 Geodetic Data Management Feature, tal como se muestra en las tablas siguientes.
- Indexa sólo el componente espacial de GeoObjects; en otras palabras, no indexa rangos de tiempo ni rangos de altitud.

Si utiliza IBM Informix Geodetic DataBlade para almacenar y manipular objetos geoespaciales en una base de datos, puede migrar los datos y las aplicaciones a IBM DB2 Geodetic Data Management Feature con algunas restricciones.

Para realizar esta tarea... :

1. Para migrar de IBM Informix Geodetic DataBlade a IBM DB2 Geodetic Data Management Feature:

*Tabla 12. Tipos de datos correspondientes de Informix Geodetic DataBlade y Geodetic Data Management Feature*

Tipo de datos de Informix Geodetic DataBlade	Tipos de datos correspondientes de DB2 Geodetic Data Management Feature	Comentarios para tipos de datos casi equivalentes
GeoEllipse		En primer lugar, conviértalo en un GeoPolygon y, a continuación, mígrelo a ST_Polygon
GeoLineseg GeoObject	ST_LineString ST_Geometry	ST_Geometry y sus subtipos no tienen soporte para los tipos de datos GeoAltRange y GeoTimeRange
GeoPoint GeoPolygon	ST_Point ST_MultiPolygon, ST_Polygon	ST_MultiPolygon requiere un punto de cierre explícito para cada anillo. Si un GeoPolygon tiene un anillo exterior, puede correlacionarse con un ST_Polygon.
GeoRing	ST_LineString	

Tabla 12. Tipos de datos correspondientes de Informix Geodetic DataBlade y Geodetic Data Management Feature (continuación)

Tipo de datos de Informix Geodetic DataBlade	Tipos de datos correspondientes de DB2 Geodetic Data Management Feature	Comentarios para tipos de datos casi equivalentes
GeoString	ST_LineString	

Los siguientes tipos de datos de Geodetic DataBlade no tienen un tipo de datos correspondiente en Geodetic Data Management Feature:

- GeoAltitude
  - GeoAltRange
  - GeoAngle
  - GeoAzimuth
  - GeoCoords
  - GeoDistance
  - GeoEllipse
  - GeoLatitude
  - GeoLongitude
  - GeoTimeRange
  - GeoVoronoi
- a. Vuelva a escribir las sentencias de SQL para utilizar DB2 los tipos de datos y funciones de Geodetic Data Management Feature.
  - b. Cargue o importe los datos en DB2 Geodetic Data Management Feature.
  - c. Vuelva a escribir las aplicaciones que utilizan Informix ODBC, ESQL/C y JDBC. En la Tabla 22 en la página 146 se muestra la conectividad de cliente correspondiente en Geodetic DataBlade and Geodetic Data Management Feature.

Tabla 13. Funciones de predicado correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature

Función en Informix Geodetic DataBlade	Función correspondiente en DB2 Geodetic Data Management Feature
Contains	ST_Contains
Inside	ST_Within
Intersect	ST_Intersects
Outside	ST_Disjoint
Within	ST_Distance

2. Las siguientes funciones de predicado de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:
  - Beyond
  - Equal
  - Nearest

Tabla 14. Funciones de producción correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature

Función en Informix Geodetic DataBlade	Función correspondiente en DB2 Geodetic Data Management Feature	Comentarios para tipos de datos casi equivalentes
Difference	ST_Difference	ST_Difference tiene soporte para puntos además de para polígonos
Generalize	ST_Generalize	
Intersection	ST_Intersection	ST_Intersection(line,line) podría producir como resultado un multipunto. ST_Intersection (line,poly) podría producir como resultado una multilínea. Devuelve el valor Empty para los objetos inconexos.
SymDifference	ST_SymDifference	ST_SymDifference tiene soporte para puntos además de para polígonos
Unión	ST_Union	ST_Union tiene soporte para puntos y para líneas además de para polígonos

Tabla 15. Funciones de accesor correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature

Función en Informix Geodetic DataBlade	Función correspondiente en DB2 Geodetic Data Management Feature	Comentarios para tipos de datos casi equivalentes
Center	ST_MidPoint, ST_PointOnSurface	ST_MidPoint puede sustituir prácticamente a líneas. ST_PointOnSurface puede sustituir prácticamente a polígonos.
Coords	ST_PointN	
Dimension	ST_Dimension	
HasZValue	ST_Is3d	
IsGeoBox	Utilice la expresión IS OF o ST_GeometryType	
IsGeoCircle	Utilice la expresión IS OF o ST_GeometryType	
IsGeoEllipse	Utilice la expresión IS OF o ST_GeometryType	
IsGeoLineseg	Utilice la expresión IS OF o ST_GeometryType	
IsGeoPoint	Utilice la expresión IS OF o ST_GeometryType	
IsGeoPolygon	Utilice la expresión IS OF o ST_GeometryType	
IsGeoRing	Utilice la expresión IS OF o ST_GeometryType	



Tabla 15. Funciones de accesor correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature (continuación)

Función en Informix Geodetic DataBlade	Función correspondiente en DB2 Geodetic Data Management Feature	Comentarios para tipos de datos casi equivalentes
IsGeoString	Utilice la expresión IS OF o ST_GeometryType	
Latitude	ST_Y	
Longitude	ST_X	
NPoints	ST_NumPoints	
NRings	ST_NumGeometries, ST_NumInteriorRing	Utilice ST_NumGeometries para obtener el número total de anillos exteriores y sume ST_NumInteriorRings para cada polígono del conjunto de multipolígonos.
Ring	ST_GeometryN, ST_ExteriorRing, ST_InteriorRingN	Utilice ST_GeometryN junto con ST_ExteriorRing y ST_InteriorRingN
SRID	ST_SRID	
Zvalue	ST_Z	

3. Las siguientes funciones de accesor de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:

- IsLarge
- IsSmallArea

Tabla 16. Funciones de modificación correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature

Función en Informix Geodetic DataBlade	Función correspondiente en DB2 Geodetic Data Management Feature
SetSRID	ST_SRID

4. Las siguientes funciones de modificación de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:

- SetAltRange
- SetAltRangeZ
- SetDist
- SetTimeRange

Tabla 17. Funciones de medida correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature

Función en Informix Geodetic DataBlade	Función correspondiente en DB2 Geodetic Data Management Feature
Area	ST_Area
Distance	ST_Distance
Length	ST_Length, ST_Perimeter

5. La función VoronoiResolution no tiene una función correspondiente en Geodetic Data Management Feature.

Tabla 18. Funciones de downcast correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature

Función en Informix Geodetic DataBlade	Función correspondiente en DB2 Geodetic Data Management Feature
GeoBox	Utilice la expresión SQL TREAT
GeoCircle	Utilice la expresión SQL TREAT
GeoEllipse	Utilice la expresión SQL TREAT
GeoLineseg	Utilice la expresión SQL TREAT
GeoPoint	Utilice la expresión SQL TREAT
GeoPolygon	Utilice la expresión SQL TREAT
GeoRing	Utilice la expresión SQL TREAT
GeoString	Utilice la expresión SQL TREAT

Tabla 19. Funciones de constructor correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature

Función en Informix Geodetic DataBlade	Función correspondiente en DB2 Geodetic Data Management Feature
GeoCoords	ST_Point
GeoPoint	ST_Point

6. Las siguientes funciones de constructor de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:

- GeoBox
- GeoCircle
- GeoEllipse
- GeoLineseg

Tabla 20. Funciones de diagnóstico correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature

Función en Informix Geodetic DataBlade	Función correspondiente en DB2 Geodetic Data Management Feature
GeoTraceLevel	Recurso de rastreo de DB2
IsValidGeometry	ST_IsValid

7. Las siguientes funciones de diagnóstico de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:

- GeoInRowSize
- GeoOutOfRowSize
- GeoRelease
- GeoTotalSize
- GeoTraceLevelSet
- GeoWarningLevel
- GeoWarningLevelSet
- IsValidSDTS

Tabla 21. Tablas de catálogo del sistema correspondientes en Informix Geodetic DataBlade y Geodetic Data Management Feature

Tabla de catálogo del sistema en Informix Geodetic DataBlade	Vista del catálogo correspondiente en DB2 Geodetic Data Management Feature
GeoLenUnit	DB2GSE.ST_UNITS_OF_MEASURE
GeoSpatialRef	DB2GSE.SPATIAL_REF_SYS

8. Las siguientes tablas de catálogo del sistema de Geodetic DataBlade no tienen una tabla o vista correspondiente en Geodetic Data Management Feature:
  - GeoEllipsoid
  - GeoParam
  - GeoVoronoi
9. Las siguientes funciones de parámetros de Geodetic DataBlade que el usuario puede establecer carecen de una función correspondiente en Geodetic Data Management Feature:
  - GeoParamSessionGet
  - GeoParamSessionSet
10. Las siguientes funciones de AltRange de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:
  - AltRange
  - Bottom
  - Contains
  - Equal
  - Inside
  - Intersect
  - IsAny
  - Outside
  - Top
11. Las siguientes funciones de TimeRange de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:
  - Begin
  - Contains
  - End
  - Equal
  - IsAny
  - Inside
  - Intersect
  - Outside
  - TimeRange
12. Las siguientes funciones de elipse de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:
  - Azimuth
  - Coords
  - Major
  - Minor
13. Las siguientes funciones de círculo de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:

- Coords
  - Radius
14. Las siguientes funciones de aritmética de ángulo de Geodetic DataBlade no tienen una función correspondiente en Geodetic Data Management Feature:
- Divide
  - Minus
  - Negate
  - Plus
  - Times
15. La siguiente conectividad de cliente de Geodetic DataBlade no tiene una conectividad de cliente correspondiente en Geodetic Data Management Feature:
- Java API
  - LIBMI

*Tabla 22. Productos con conectividad de cliente correspondiente en Geodetic DataBlade y DB2 Geodetic Data Management Feature*

Conectividad de cliente en Informix Geodetic DataBlade	Conectividad de cliente correspondiente en DB2 Geodetic Data Management Feature
ESQLC	SQLC
ODBC	ODBC
JDBC	JDBC

---

## Cómo llenar datos espaciales con datos geodésicos

Después de crear columnas espaciales y registrar aquellas en las que piensa crear un índice espacial, está preparado para llenar las columnas con datos geodésicos. Puede proporcionar datos geodésicos importándolos en formato Shape o bien insertando o actualizando valores en los formatos de datos siguientes:

- Forma
- Texto convencional (WKT)
- Binario convencional (WKB)
- GML (Geography Markup Language)

### Restricción:

- Con Spatial Extender, no puede utilizar los mandatos del geocodificador ni procedimientos almacenados para convertir datos en datos geodésicos.
- Para el comportamiento geodésico, utilice sistemas de referencia espacial que tengan SRID comprendidos entre 2.000.000.000 y 2.000.001.000.
- Los datos de formas deben estar en un sistema de coordenadas geográficas.

El procedimiento para importar datos geodésicos es el mismo que se utiliza con datos espaciales.

---

## Capítulo 18. Índices geodésicos

Puede crear índices Voronoi geodésicos que mejoren el rendimiento de las consultas de datos geodésicos. Este capítulo:

- Describe los índices Voronoi geodésicos
- Describe las estructuras de celdas de Voronoi e indica cuándo puede seleccionar una estructura alternativa.
- Explica el modo de crear un índice Voronoi geodésico.

---

### Índices Voronoi geodésicos

DB2® Geodetic Data Management Feature proporciona un índice Voronoi que agiliza la velocidad el acceso a los geodésicos. Este índice organiza el acceso a los datos geodésicos utilizando formaciones teseladas Voronoi de la superficie de la tierra.

Geodetic Data Management Feature calcula el círculo delimitador mínimo (MBC) para cada geometría. MBC es un círculo que rodea una geometría geodésica. El índice Voronoi utiliza esta información de MBC para organizar los datos en una estructura de celdas. Una búsqueda que utiliza un índice Voronoi puede descender rápidamente por los datos organizados para encontrar objetos en el área de interés general y, a continuación, realizar pruebas más exactas en los propios objetos. Un índice Voronoi puede mejorar el rendimiento porque elimina la necesidad de examinar objetos situados fuera del área de interés. Sin un índice Voronoi, una consulta tendría que evaluar cada objeto para encontrar aquellos que coincidan con los criterios de búsqueda.

El optimizador considera que un índice Voronoi lo utilizarán todas las consultas que contienen las siguientes funciones en su cláusula WHERE:

- EnvelopesIntersect
- ST\_Contains
- ST\_Distance
- ST\_EnvIntersects
- ST\_Intersects
- ST\_MBRIntersects
- ST\_Within

Cuando cree un índice Voronoi geodésico, puede elegir una estructura de celdas de Voronoi alternativa.

---

### Estructuras de celdas de Voronoi

Para realizar cálculos eficaces, DB2® Geodetic Data Management subdivide la superficie de la tierra, en celdas más pequeñas y fáciles de gestionar, parecidas a las de un panal de abejas. Esta subdivisión se llama *formación teselada Voronoi*, y la estructura de datos que la describe se llama *estructura de celdas de Voronoi*. Una formación teselada Voronoi es una estructura de celdas en la que el interior de cada celda está compuesto por todos los puntos cercanos a un punto del entramado particular, más que a cualquier otro punto. En una estructura de celdas

de Voronoi, las celdas son *envolturas convexas*. Una envoltura convexa de un conjunto de puntos es el conjunto convexo más pequeño que incluye los puntos (o el polígono más pequeño que define el "exterior" de un grupo de puntos). Las estructuras de celdas de Voronoi tienden a ser polígonos de forma irregular; el número y la ubicación de las celdas pueden ajustarse para coincidir con la densidad y la ubicación de los datos espaciales.

Por ejemplo, una estructura de celdas de Voronoi puede subdividir la tierra en polígonos en función de la población humana. Allá donde la población y los datos sean densos aparecerán polígonos pequeños. Allá donde la población sea escasa aparecerán polígonos grandes.

La Figura 21 muestra la estructura Voronoi que se basa en la densidad de población mundial. Geodetic Data Management Feature utiliza esta estructura de celdas en sus cálculos espaciales.

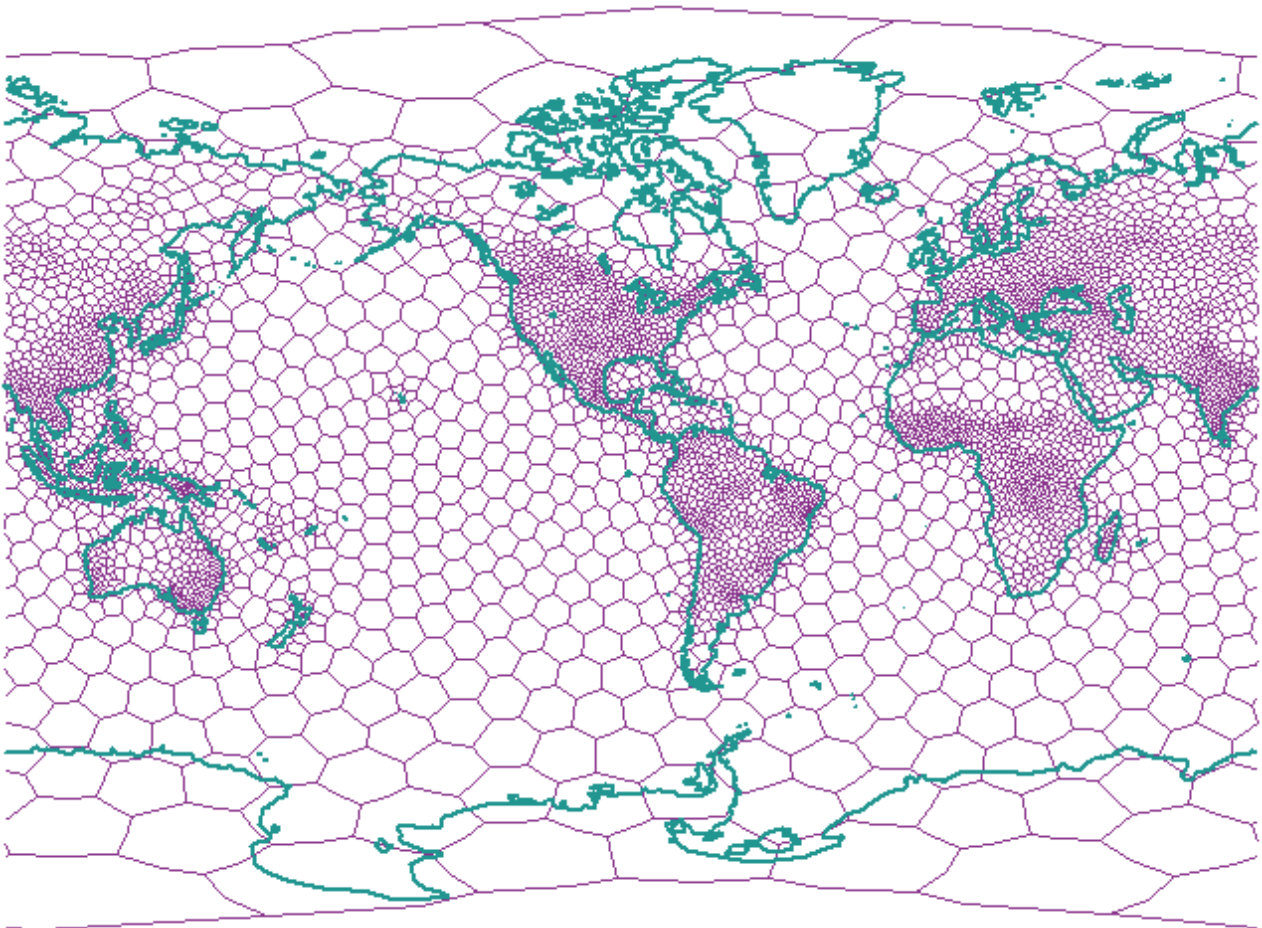


Figura 21. Estructura Voronoi basada en la densidad de población mundial

---

## Consideraciones para seleccionar una estructura de celdas de Voronoi alternativa

Todas las operaciones realizadas con geometrías geodésicas utilizan un ID de Voronoi de 1 que especifica la estructura de celdas de Voronoi basada en la densidad de población mundial. Al crear un índice, si los datos están agrupados en una o varias áreas de la tierra, como los datos sobre calles de uno o varios países, puede elegir una estructura de celdas de Voronoi alternativa que tiene celdas más pequeñas en las áreas en que están ubicados los datos (porque la resolución es inversamente proporcional al tamaño de las celdas). DB2<sup>®</sup> Geodetic Data Management Feature proporciona un número de estructuras de celdas de Voronoi para indexar que podrían ajustarse a los datos.

### **Restricción:**

Sólo puede elegir una estructura de celdas alternativa cuando cree un índice Voronoi geodésico.

La estructura dodeca04 (ID de Voronoi 12) es adecuada para datos que están distribuidos uniformemente sobre toda la superficie de la tierra, como las imágenes por satélite. Todas las celdas son más o menos uniformes en cuanto al tamaño y la resolución, en el peor de los casos, es de 10 centímetros aproximadamente. Considere la posibilidad de utilizar una estructura de celdas de Voronoi distinta de la estructura de la población mundial por omisión (ID de Voronoi 1) o de la estructura dodeca04 si alguna de las siguientes condiciones se aplica a sus datos o a su aplicación:

### **Alta resolución**

Si necesita determinar si forman intersección los objetos separados por menos de 10 centímetros, deberá utilizar una estructura de celdas de Voronoi que tenga celdas más pequeñas en las regiones donde se ubican sus datos. La resolución es inversamente proporcional al tamaño de las celdas.

### **Polígonos con muchos vértices**

Si los datos están formados por polígonos que tienen un número relativamente grande de vértices y su superficie es relativamente pequeña, se recomienda utilizar en su lugar una estructura de celdas de Voronoi que tenga más celdas en las regiones de interés. Si la mayoría de los polígonos tienen 50 vértices o menos, probablemente no deseará hacerlo. Si los únicos polígonos del conjunto de datos que tienen muchos vértices tienen tamaño de continente, probablemente tampoco deseará hacerlo.

Si tiene muchos polígonos de 3000 vértices con un tamaño como el de un condado de EE.UU., podrá mejorar significativamente el rendimiento de la consulta utilizando una estructura de celdas distinta, especialmente si la aplicación realiza varias consultas de polígono-forma intersección-polígono.

### **Datos muy densos**

Si los datos se concentran en regiones muy pequeñas (por ejemplo, si tiene cientos de objetos por kilómetro cuadrado) podrá mejorar el rendimiento de la consulta utilizando una estructura de celdas de Voronoi cuya densidad de celdas coincida con la densidad de los datos.

---

## Creación de índices Voronoi geodésicos

Requisitos previos

Para crear un índice Voronoi geodésico, el ID de usuario debe tener las mismas autorizaciones y privilegios que cuando creó un índice reticular espacial.

**Restricción:**

Las mismas restricciones para crear índices con la sentencia CREATE INDEX están en vigor cuando se crea un índice Voronoi geodésico. Es decir, la columna en la que se crea un índice debe ser una columna de tabla base no una columna de vista ni una columna de seudónimo. El sistema de base de datos de DB2 resolverá los alias en el proceso.

DB2 Geodetic Data Management Feature proporciona un nuevo método de acceso espacial que le permite crear índices en columnas que contengan datos geodésicos. Las consultas que utilizan un índice se pueden ejecutar más rápidamente.

Puede crear un índice Voronoi geodésico de cualquiera de las maneras siguientes:

- Utilice la ventana Crear índice del Centro de control de DB2.
- Utilice la sentencia CREATE INDEX de SQL con la extensión db2gse.spatial\_index en la cláusula EXTEND USING.

Cree un índice Voronoi geodésico utilizando el Centro de control de DB2 o el procesador de línea de mandatos.

- Para crear un índice Voronoi geodésico utilizando el Centro de control, pulse con el botón derecho del ratón en la tabla que tiene la columna espacial en la que desea crear un índice Voronoi geodésico y pulse **Spatial Extender —? Índices espaciales** desde el menú. Se abrirá la ventana Índices espaciales. Siga los indicadores de mandatos para completar la tarea.
- Para crear un índice Voronoi geodésico utilizando la sentencia SQL CREATE INDEX, emita la sentencia CREATE INDEX utilizando la cláusula EXTEND USING y la extensión de índice reticular db2gse.spatial\_index.

En el ejemplo siguiente, la sentencia CREATE INDEX crea el índice geodésico STORESX1 en la columna espacial LOCATION en la tabla CUSTOMERS:

```
CREATE INDEX storesx1
  ON customers (location)
  EXTEND USING db2gse.spatial_index (-1, -1, 1)
```

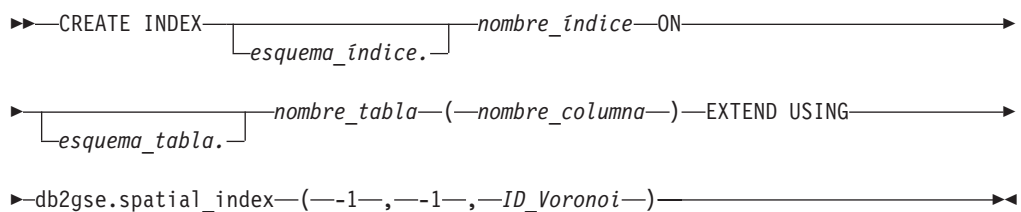
Para un índice geodésico Voronoi, debe especificar el valor -1 en los dos primeros parámetros de la cláusula USING db2gse.spatial\_index.

---

## Sentencia CREATE INDEX para un índice Voronoi geodésico

Utilice la sentencia CREATE INDEX con la cláusula EXTEND USING para crear un índice Voronoi geodésico.

**Sintaxis:**



Donde:



**esquema\_índice**

Nombre del esquema al que debe pertenecer el índice que está creando. Si no especifica un nombre, el sistema de base de datos de DB2 utiliza el nombre de esquema que se ha almacenado en el registro especial CURRENT SCHEMA.

**nombre\_índice**

Nombre, no calificado, del índice geodésico que está creando.

**esquema\_tabla**

Nombre del esquema al que pertenece la tabla donde reside *nombre\_columna*. Si no especifica un nombre, el sistema de base de datos de DB2 utiliza el nombre de esquema que se ha almacenado en el registro especial CURRENT SCHEMA.

**nombre\_tabla**

Nombre, no calificado, de la tabla donde reside *nombre\_columna*.

**nombre\_columna**

Nombre de la columna espacial en la que se ha creado el índice Voronoi geodésico.

**ID\_Voronoi**

Un número entero que identifica el ID de la estructura de celdas de Voronoi. Se encuentran disponibles catorce estructuras de celdas de Voronoi. Un ID de Voronoi de 1 especifica la estructura de celdas Voronoi basada en la densidad de población mundial que DB2 Geodetic Data Management Feature también utiliza para todas las operaciones espaciales.

**Ejemplos**

En el siguiente ejemplo, la sentencia CREATE INDEX crea el índice geodésico STORESX1 en la columna espacial LOCATION de la tabla CUSTOMERS:

```
CREATE INDEX storesx1
  ON customers (location)
  EXTEND USING db2gse.spatial_index (-1, -1, 1)
```

El optimizador considera que un índice Voronoi lo utilizarán todas las consultas que contienen las siguientes funciones en su cláusula WHERE:

- ST\_Contains
- ST\_Distance
- ST\_Intersects
- ST\_MBRIntersects
- ST\_EnvIntersects
- EnvelopesIntersect
- ST\_Within

Las siguientes sentencias muestran el uso de un índice Voronoi. En primer lugar, inserte datos en la tabla CUSTOMER. Puede entrar los valores directamente, tal como se muestra en esta primera sentencia INSERT:

```
INSERT INTO customer
(id, last_name, first_name, address, city, state, zip,
location)
VALUES
('123-456789', 'Duck', 'Donald',
'123 Mallard Way', 'Wetland Marsh', 'ND', '55555-5555',
db2gse.ST_GeomFromWKT('POINT(123.123, 45.67)', 2000000000))
```

De forma alternativa, puede utilizar variables en una aplicación, tal como muestra la siguiente consulta, para insertar valores en una tabla:

```
INSERT INTO customer
(id, last_name, first_name,
address, city, state, zip,
location)
VALUES
(:mid, :mlast, :mfirst,
:address, :mcity, :mstate, :mzip,
db2gse.ST_GeomFromWKB(:mlocation))
```

La siguiente sentencia UPDATE modifica los datos insertados. No utiliza el índice STORESX1 porque no utiliza las funciones ST\_Contains, ST\_Distance, ST\_Intersects, ST\_MBRIntersects, ST\_EnvIntersects, EnvelopesIntersect ni ST\_Within en su cláusula WHERE.

```
UPDATE customer
SET location = db2gse.ST_GeomFromWKT('POINT(123.123, 45.67)',
2000000000)
WHERE id = '123-456789';
```

Las siguientes sentencias DELETE pueden utilizar el índice STORESX1, si el optimizador determina que el índice mejora el rendimiento, porque las sentencias DELETE utilizan la función ST\_Within y las funciones ST\_Intersects en sus cláusulas WHERE respectivamente:

```
DELETE FROM customers
WHERE db2gse.ST_Within(location, :BayArea) = 1;
DELETE FROM customers
WHERE db2gse.ST_Intersects(c.location, :BayArea) = 1
```

Las dos sentencias SELECT siguientes también pueden utilizar el índice STORESX1:

```
SELECT s.id, AVG(c.location..ST_Distance(s.location))
FROM customers c, stores s
WHERE db2gse.ST_Within(c.location, s.zone) = 1
GROUP BY s.id;
SELECT c.location..ST_AsText()
FROM customers c
WHERE db2gse.ST_Within(c.location, :BayArea) = 1
```

---

## Estructuras de celdas Voronoi suministradas con DB2 Geodetic Data Management Feature

Cada estructura de celdas de Voronoi cubre toda la tierra. En las ilustraciones siguientes, sólo se muestran aquellas porciones de la tierra contenidas en el área en que las celdas son densas para dicha estructura de celdas Voronoi. Cuando seleccione una estructura de celdas de Voronoi, tenga en cuenta que las celdas situadas fuera de las áreas ilustradas serán grandes y su resolución será, como corresponde, inferior. Si los datos se encuentran en dichas áreas escasas, es posible que se reduzca el rendimiento de la consulta.

La tabla siguiente lista las estructuras de celdas Voronoi que suministra DB2 Geodetic Data Management Feature. Dichas estructuras de celdas de Voronoi las proporciona Geodyssey Ltd.

Tabla 23. Estructuras de celdas Voronoi

Descripción	ID de Voronoi	Ilustración
Mundo, basado en la densidad de población	1	Figura 22 en la página 154
Estados Unidos	2	Figura 23 en la página 155
Canadá	3	Figura 24 en la página 156
India	4	Figura 25 en la página 157
Japón	5	Figura 26 en la página 158
África	6	Figura 27 en la página 159
Australia	7	Figura 28 en la página 160
Europa	8	Figura 29 en la página 161
Norteamérica	9	Figura 30 en la página 162
Sudamérica	10	Figura 31 en la página 163
Mediterráneo	11	Figura 32 en la página 164
Mundo, distribución de datos uniforme, resolución media (dodeca04)	12	Figura 33 en la página 165
Mundo, basada en la producción industrial (naciones del G7)	13	Figura 34 en la página 166
Mundo, distribución de datos uniforme, baja resolución (isotipo)	14	Figura 35 en la página 167

## Mundo, basado en la densidad de población (ID de Voronoi: 1)

Las celdas de Voronoi subdividen la tierra basándose en la densidad de la población mundial.

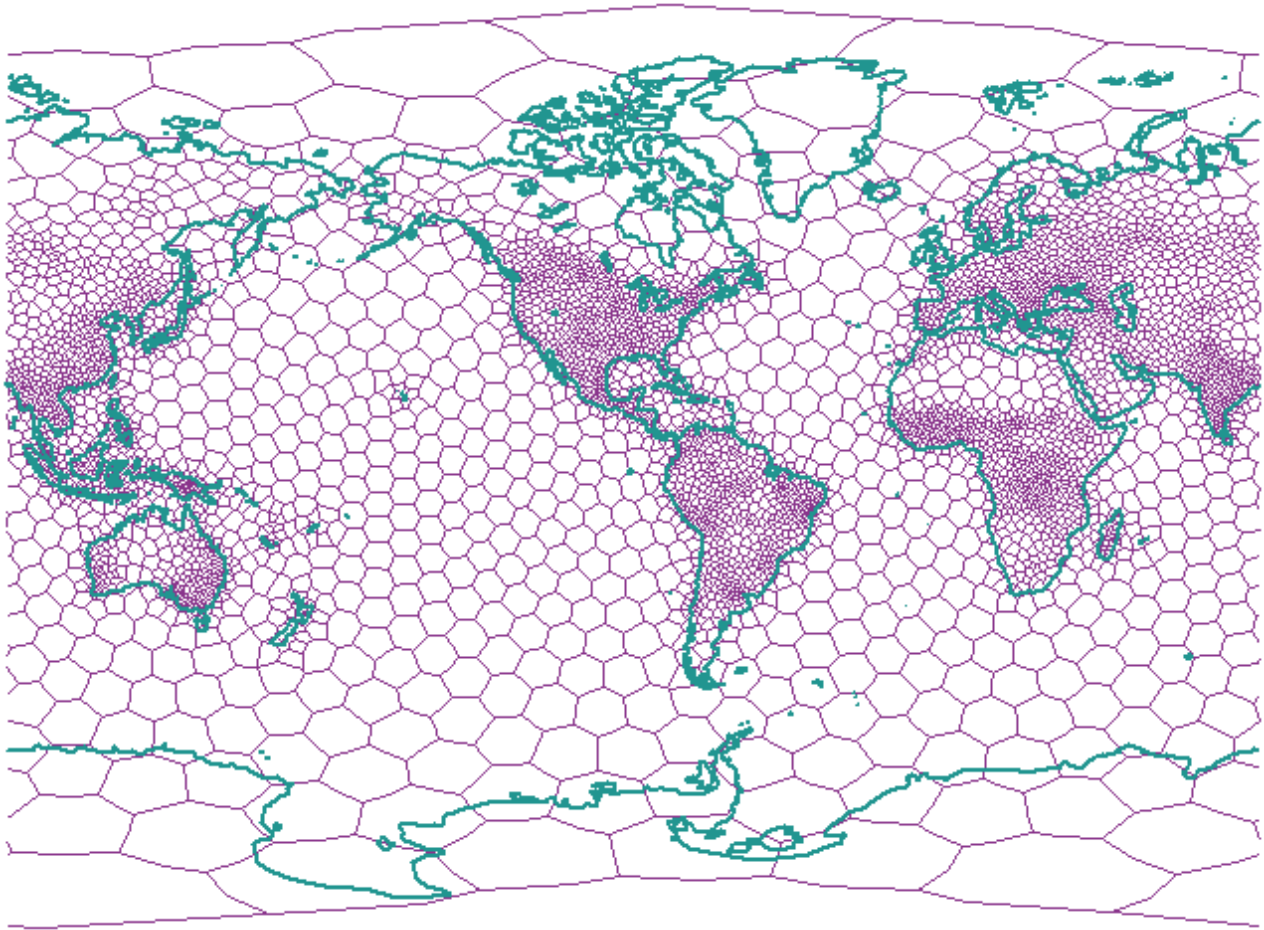


Figura 22. Estructura de celdas de Voronoi para el mundo (población)

### **Estados Unidos (ID de Voronoi: 2)**

Las celdas de Voronoi subdividen los Estados Unidos basándose en la densidad de la población.

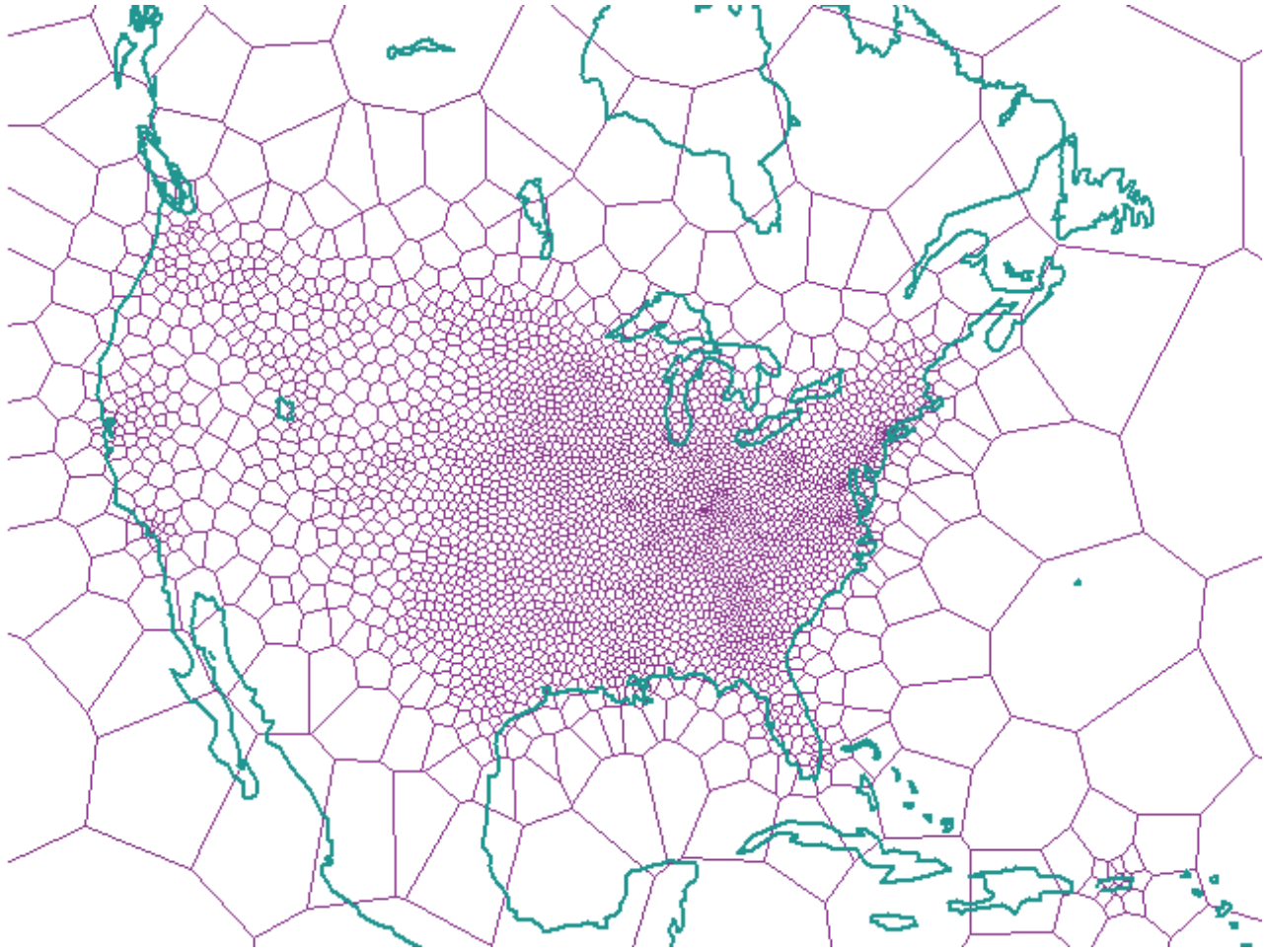


Figura 23. Estructura de celdas de Voronoi para los EE.UU.

### Canadá (ID de Voronoi: 3)

Las celdas de Voronoi subdividen Canadá basándose en la densidad de población.

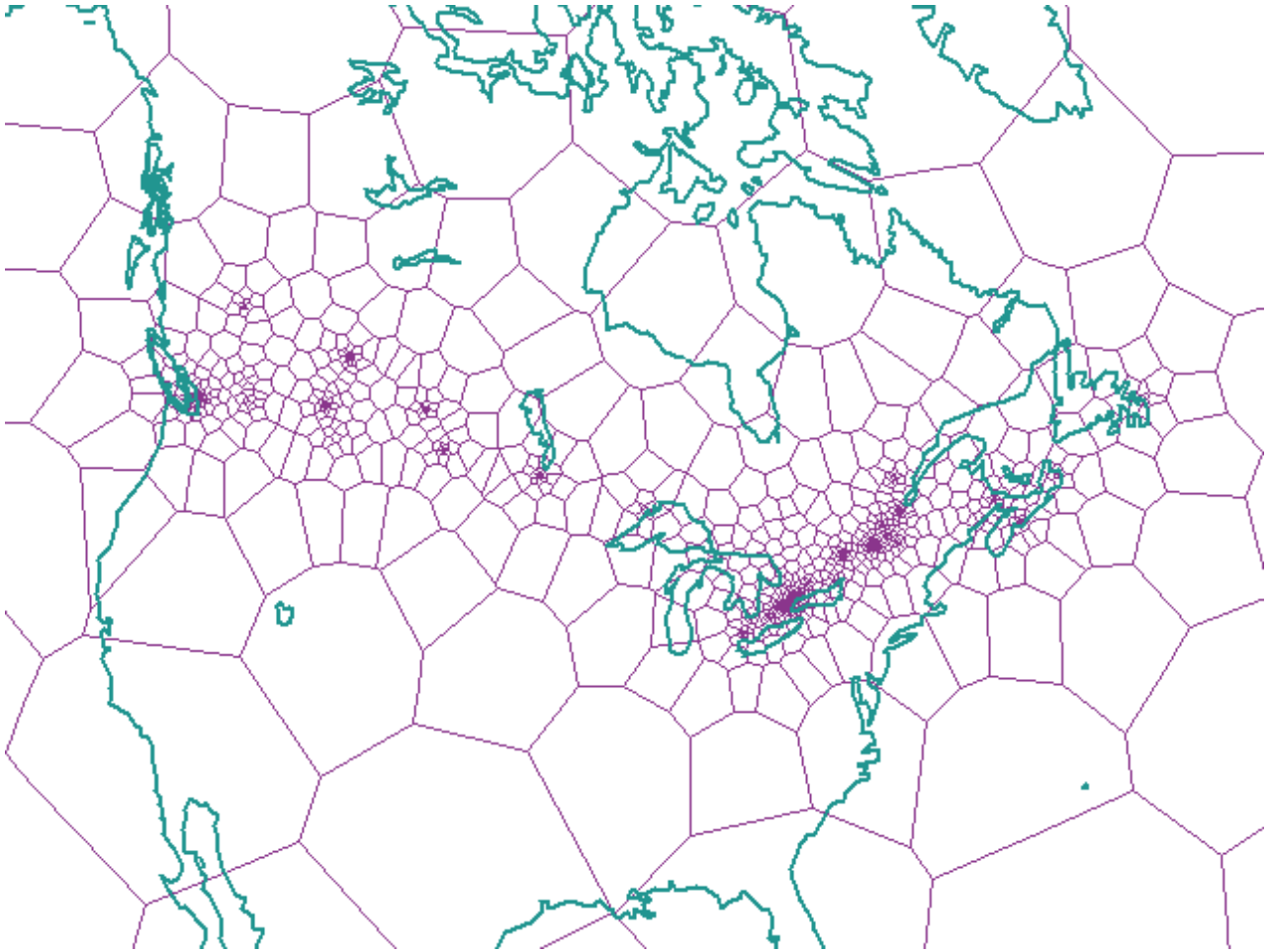


Figura 24. Estructura de celdas de Voronoi para Canadá

## India (ID de Voronoi: 4)

Las celdas de Voronoi subdividen la India basándose en la densidad de población.

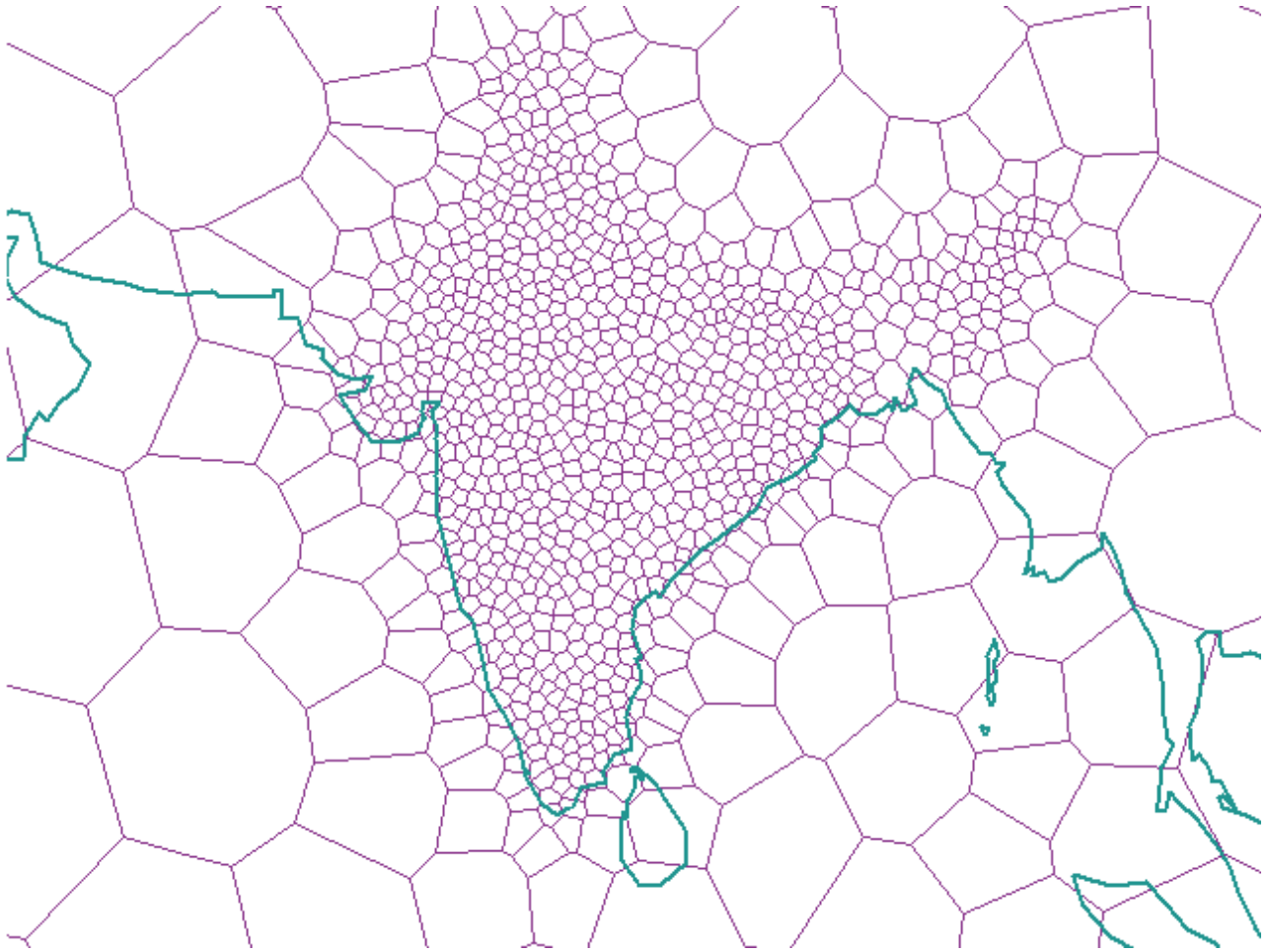


Figura 25. Estructura de celdas de Voronoi para la India

## Japón (ID de Voronoi: 5)

Las celdas de Voronoi subdividen Japón basándose en la densidad de población.

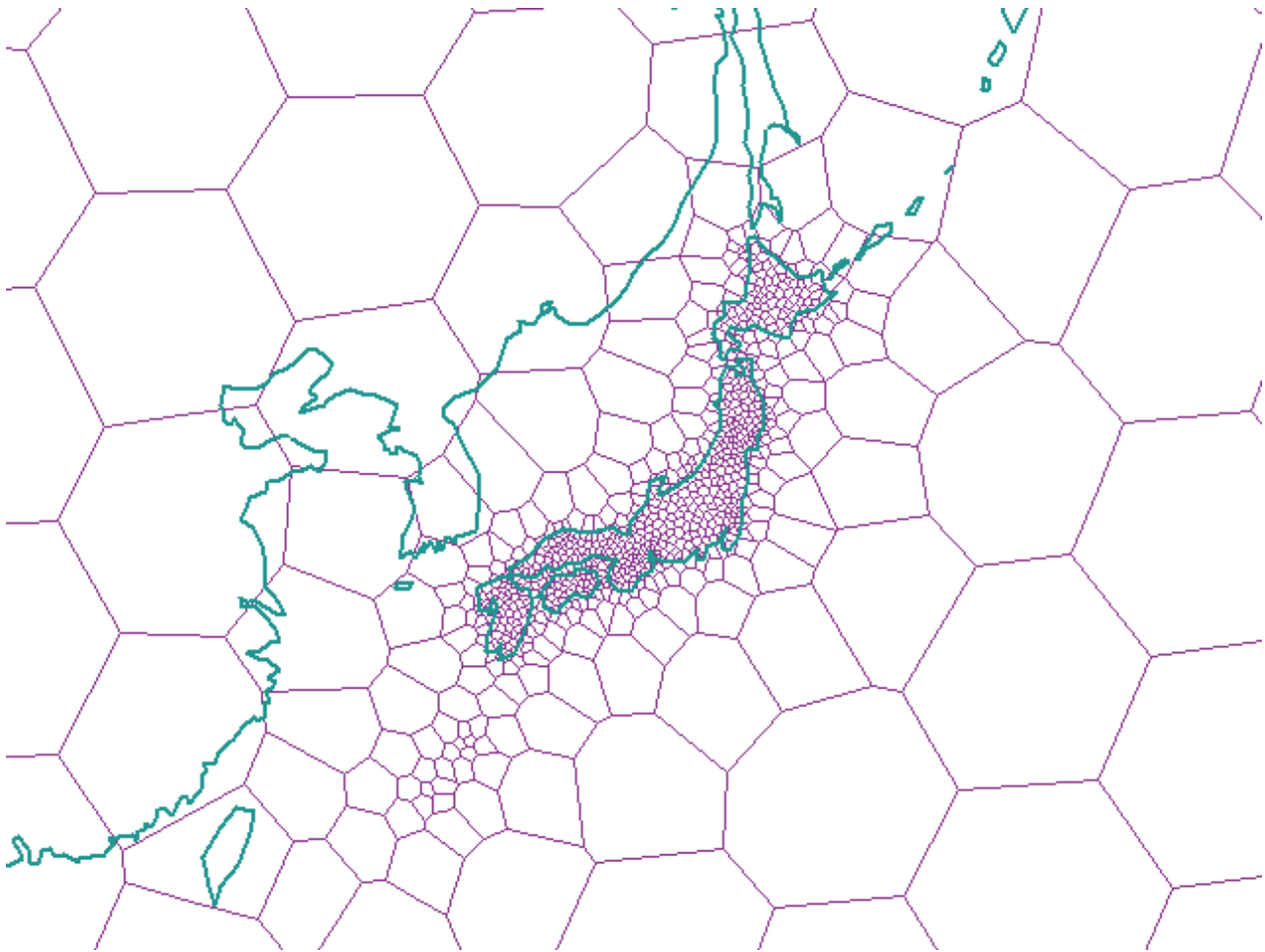


Figura 26. Estructura de celdas de Voronoi para Japón



## Africa (ID de Voronoi: 6)

Las celdas de Voronoi subdividen África basándose en la densidad de población.

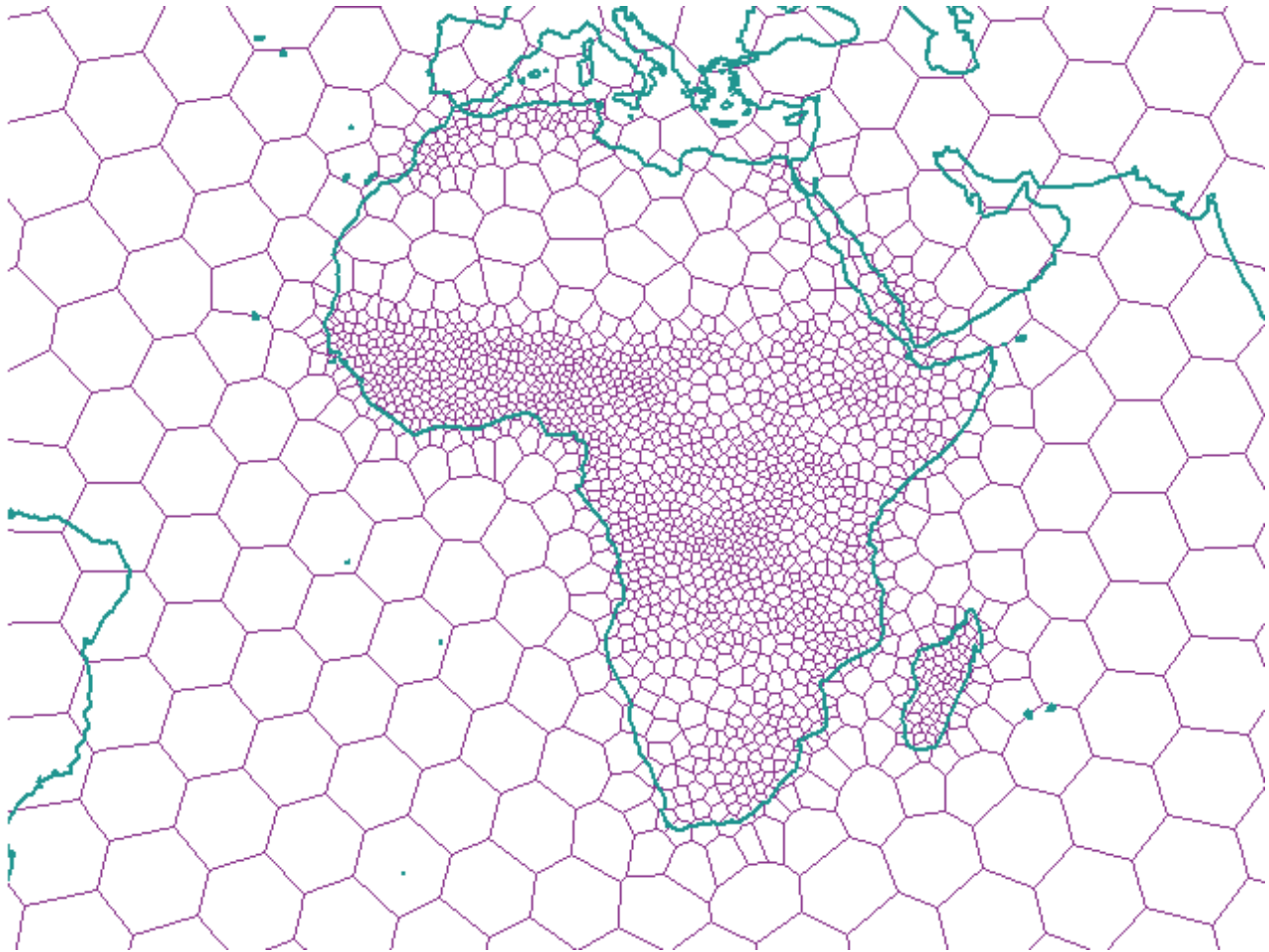


Figura 27. Estructura de celdas de Voronoi para África

## Australia (ID de Voronoi: 7)

Las celdas de Voronoi subdividen Australia basándose en la densidad de población.

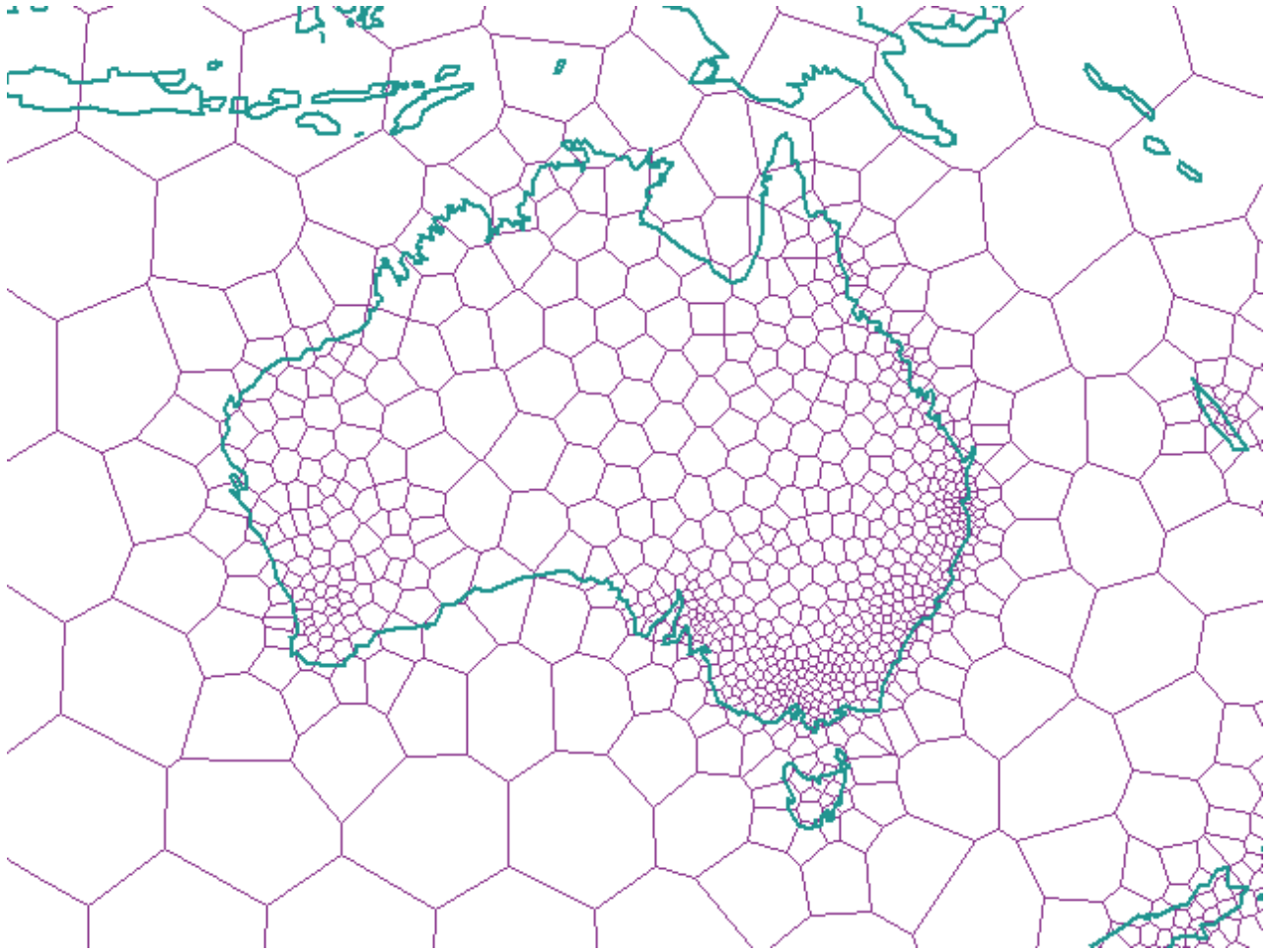


Figura 28. Estructura de celdas de Voronoi para Australia

## Europa (ID de Voronoi: 8)

Las celdas de Voronoi subdividen Europa basándose en la densidad de población.

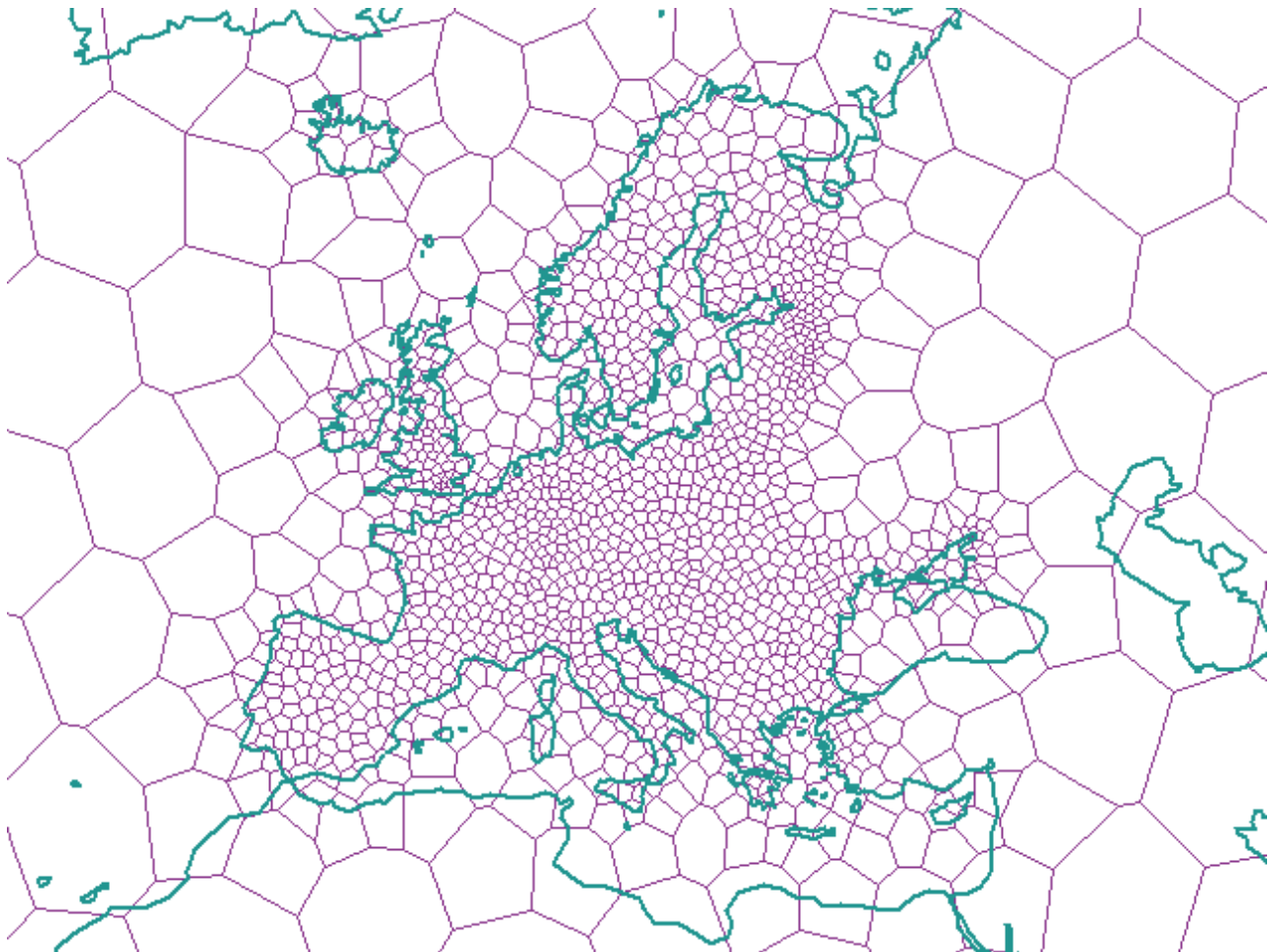


Figura 29. Estructura de celdas de Voronoi para Europa

## Norteamérica (ID de Voronoi: 9)

Las celdas de Voronoi subdividen Norteamérica basándose en la densidad de población.

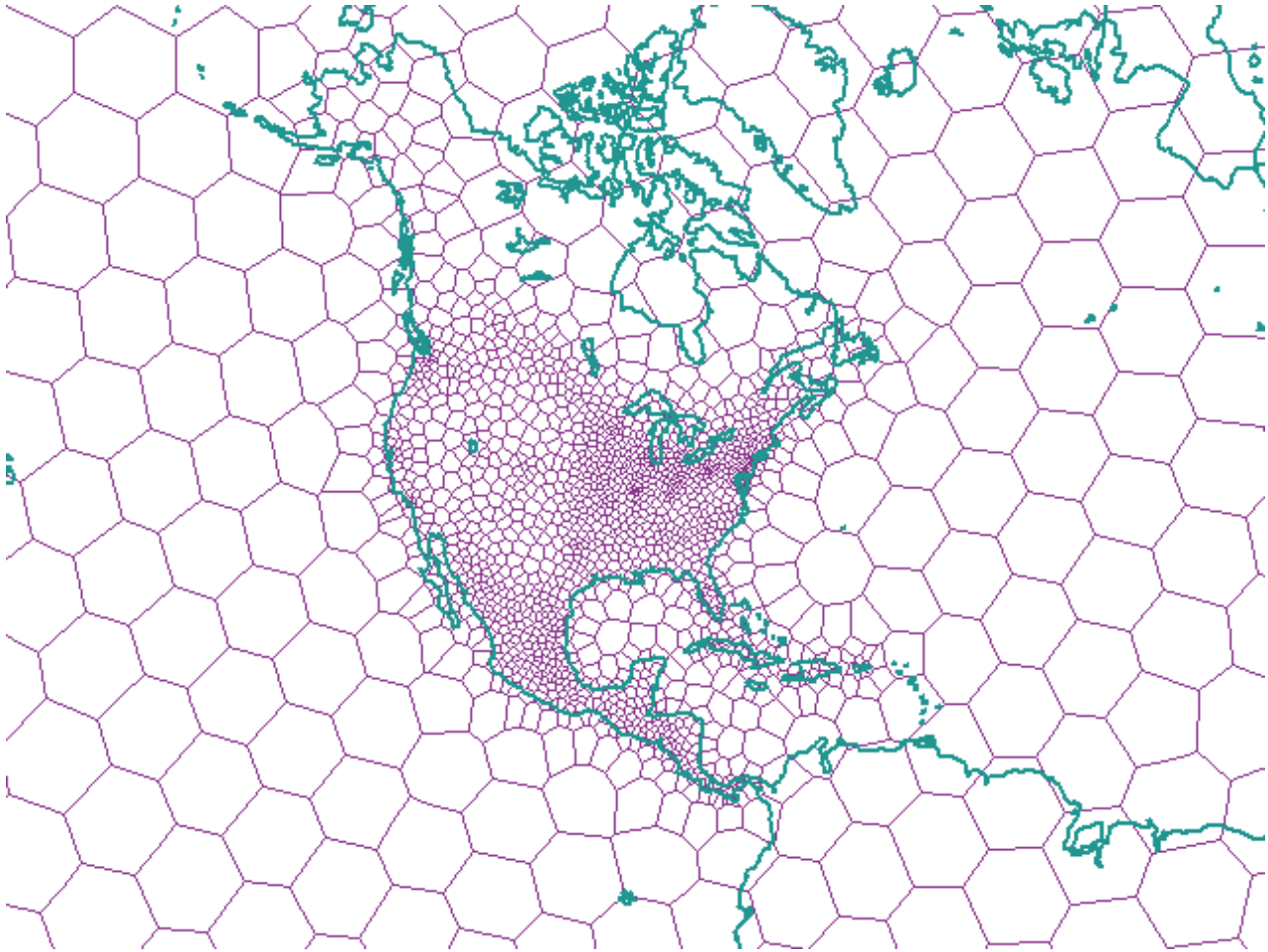


Figura 30. Estructura de celdas de Voronoi para Norteamérica

### **Sudamérica (ID de Voronoi: 10)**

Las celdas de Voronoi subdividen Sudamérica basándose en la densidad de población.



Figura 31. Estructura de celdas de Voronoi para Sudamérica

### **Mediterráneo (ID de Voronoi: 11)**

Las celdas de Voronoi subdividen el área del Mediterráneo basándose en la densidad de población.

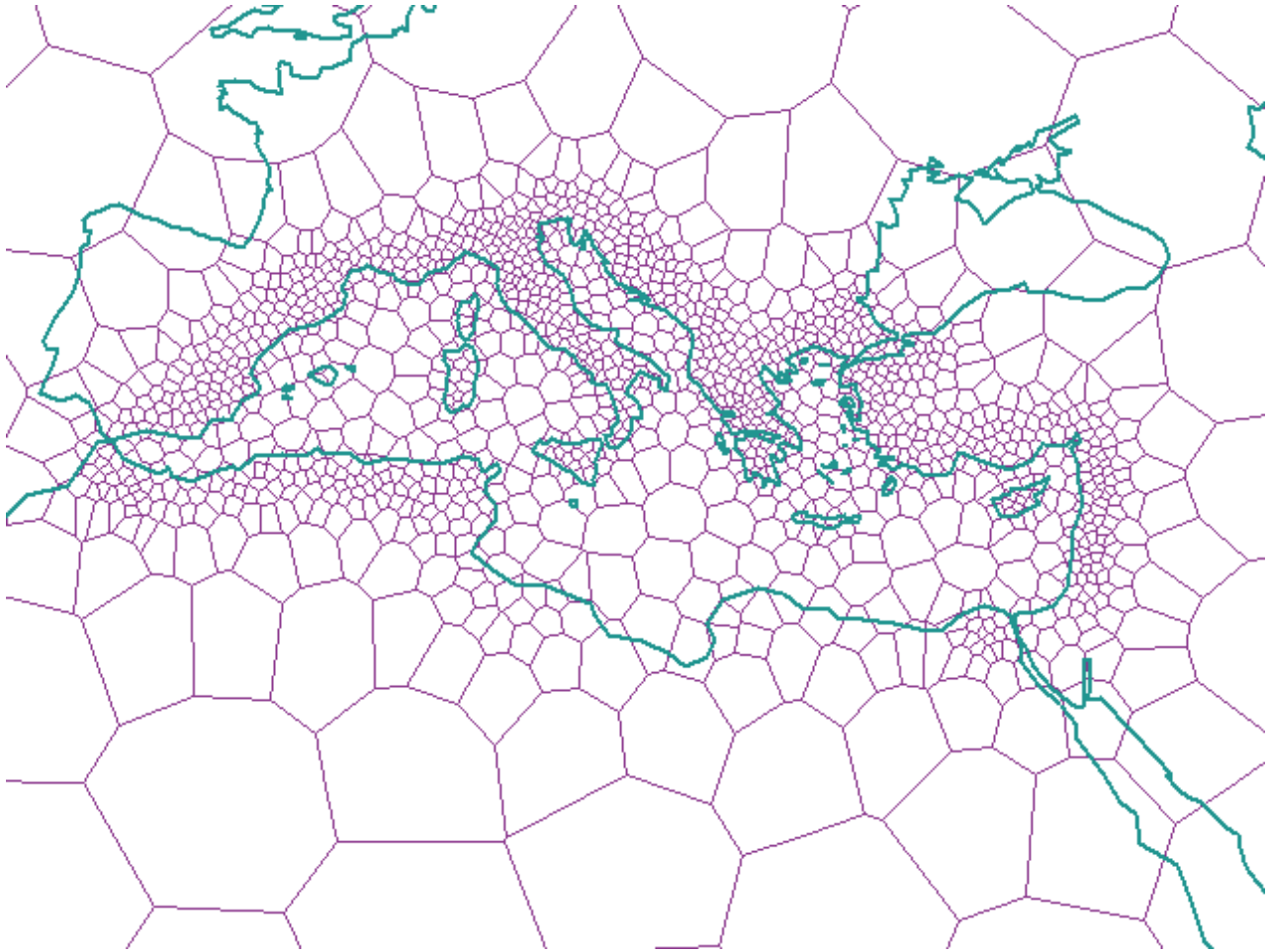


Figura 32. Estructura de celdas de Voronoi para el área del Mediterráneo

**Mundo, distribución uniforme de datos, resolución media:  
dodeca04 (ID de Voronoi: 12)**

Las celdas de Voronoi subdividen el mundo con una distribución uniforme de datos y una resolución media.



Figura 33. Estructura de celdas de Voronoi para el mundo (dodeca04)

### **Mundo, países industrializados: países del G7 (ID de Voronoi: 13)**

Las celdas de Voronoi subdividen el mundo basándose en la producción industrial de sus naciones.

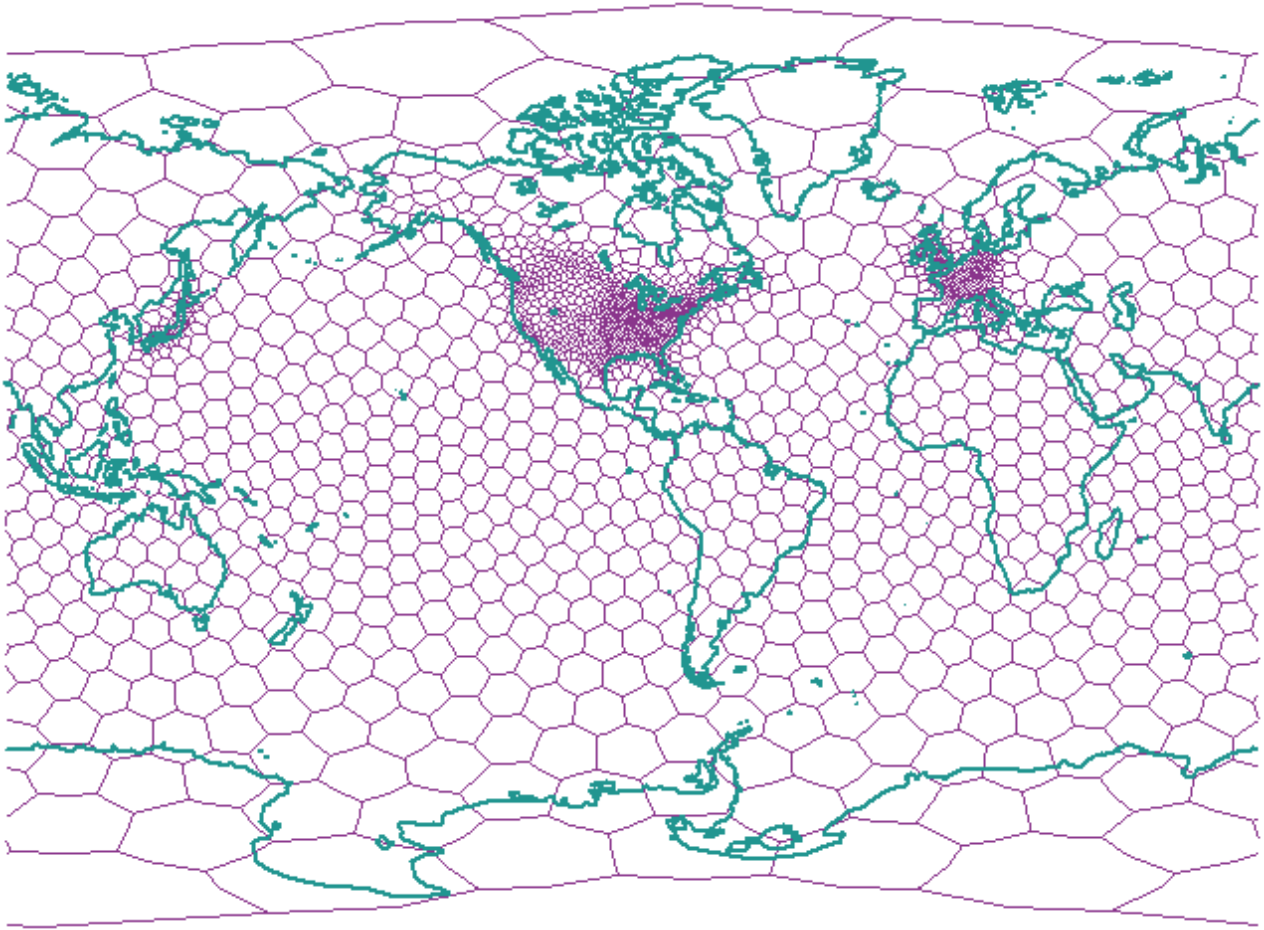


Figura 34. Estructura de celdas de Voronoi para (naciones del g7)

**Mundo, distribución uniforme de datos, resolución baja:  
isotipo (ID de Voronoi: 14)**

Las celdas de Voronoi subdividen el mundo con una distribución uniforme de datos y una resolución baja.





*Figura 35. Estructura de celdas de Voronoi para el mundo (isotipo)*



---

## Capítulo 19. Diferencias en la utilización de datos geodésicos y espaciales

Este capítulo describe las siguientes diferencias en la utilización de datos geodésicos y espaciales:

- Atributos *x* e *y* mínimos y máximos para tipos de datos `ST_Geometry`
- Diferencias entre la utilización de representaciones terrestres y planas
- Funciones espaciales soportadas por DB2 Geodetic Data Management Feature y diferencias en el comportamiento de las funciones
- Procedimientos almacenados y vistas de catálogo soportados por DB2 Geodetic Data Management Feature
- Sistemas de referencia espacial geodésicos adicionales y elipsoides geodésicos

---

### Atributos *x* e *y* mínimos y máximos

DB2® Geodetic Data Management Feature utiliza un círculo delimitador mínimo (MBC) en lugar de un rectángulo delimitador mínimo para organizar los datos en estructuras de celdas para el índice Voronoi geodésico.

En las geometrías geodésicas, el MBC es un círculo que rodea las geometrías, y los atributos *x* e *y* mínimos y máximos tienen los siguientes valores internos:

- xmin** El término *i* del coseno de dirección del centro del círculo delimitador.
- xmax** El término *j* del coseno de la dirección del centro del círculo delimitador.
- ymin** El término *k* del coseno de la dirección del centro del círculo delimitador.
- ymax** El *radio\_arco* del círculo delimitador.

En las geometrías geodésicas, las funciones `ST_MinX`, `ST_MaxX`, `ST_MinY` y `ST_MaxY` muestran puntos a lo largo del MBC. Los resultados de estas funciones todavía generan valores de longitud y latitud similares a las geometrías espaciales, pero los resultados pueden diferir en las geometrías geodésicas de la siguiente manera:

- Si el MBC cruza la línea de datos, el valor de `ST_MinX` es mayor que el valor de `ST_MaxX`. Por ejemplo, si el centro de un MBC se encuentra en la línea de datos y tiene un radio de 5 grados, el valor de `ST_MinX` es 175 y el valor de `ST_MaxX` es -175.
- Si el MBC incluye el polo norte o el polo sur, `ST_MinX` es -180 y `ST_MaxX` es 180.
- Si el MBC incluye el polo norte, el valor de `ST_MaxY` es 90.
- Si el MBC incluye el polo sur, el valor de `ST_MinY` es -90.

---

### Diferencias entre la utilización de representaciones terrestres y planas

DB2 Spatial Extender y DB2 Geodetic Data Management Feature utilizan tecnologías básicas distintas:

- Spatial Extender utiliza un mapa plano (o planar) basado en coordenadas proyectadas. Sin embargo, la proyección en un mapa no es capaz de representar fielmente toda la tierra porque los mapas tienen bordes y la tierra carece de ellos.
- Geodetic Data Management Feature utiliza un elipsoide como modelo y considera que la tierra es un globo sin fisuras que no tiene distorsiones en los polos ni en los bordes en el meridiano 180°.

En esta sección, el término "Tierra plana" hace referencia al uso de una proyección para representar la Tierra completa. El término "Tierra redonda" hace referencia al uso de un sistema de referencia donde se utiliza un elipsoide como modelo terrestre.

Las distintas tecnologías producen diferencias en el modo en que las geometrías se manejan en determinadas situaciones, especialmente las que se ilustran en este tema:

- Segmentos lineales (y distancias medidas) que cruzan el meridiano 180°.
- Polígonos que ocupan el meridiano 180°.
- Rectángulos delimitadores mínimos que cruzan el meridiano 180°.
- Polígonos que delimitan un polo.
- Polígonos que representan hemisferios, cinturones ecuatoriales o toda la tierra.

Geodetic Data Management Feature ofrece ventajas exclusivas al trabajar con geometrías que atraviesan el meridiano 180° o que están cercanas a un polo, donde la representación de tierra plana que utiliza Spatial Extender encuentra limitaciones.

## Segmentos lineales que cruzan el meridiano número 180

Una representación de la tierra redonda proporciona una vía de acceso más corta que una proyección de la tierra plana.

La Figura 36 en la página 171 muestra las distintas formas que Spatial Extender y Geodetic Data Management Feature manejan un segmento lineal que cruza el meridiano número 180. En este ejemplo, se utiliza el segmento lineal para calcular la distancia entre Anchorage y Tokio. Geodetic Data Management Feature mide la distancia entre dos puntos a lo largo de una geodésica, la vía de acceso más corta entre dos puntos del elipsoide. Los dos puntos pueden estar ubicados en cualquier lugar del globo, y Geodetic Data Management Feature elige correctamente un segmento lineal que transcurre en dirección oeste desde Anchorage hasta Tokio porque utiliza la representación terrestre redonda. Sin embargo, puesto que Spatial Extender utiliza una proyección de mapa plano, Spatial Extender ignora que un segmento lineal podría conectar Anchorage y Tokio de esa manera, por lo que elige un segmento lineal mucho más largo que transcurre en dirección este hasta Tokio. La proyección de mapa plano tiene el meridiano -180° en el borde izquierdo y el meridiano 180° en el borde derecho.

Para obtener un resultado correcto utilizando Spatial Extender deberá realizar una de las siguientes acciones:

- Dividir el segmento lineal en dos segmentos lineales, uno al este del meridiano 180° y el otro al oeste del mismo.
- Volver a proyectar los datos de modo que el meridiano 180° no esté en un borde.

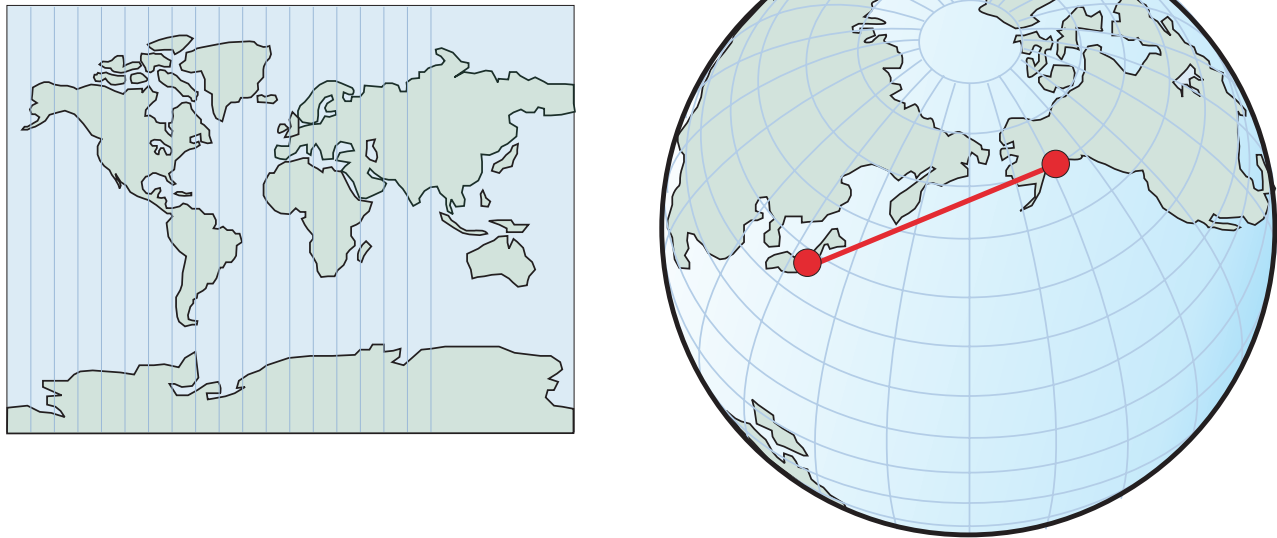


Figura 36. Líneas que cruzan el meridiano 180°

## Polígonos que ocupan el meridiano 180°

Para trabajar con un polígono que ocupa el meridiano 180°, la representación terrestre plana (Spatial Extender) requiere que el usuario divida el polígono en dos partes.

En el siguiente ejemplo se muestra un polígono para la parte que se encuentra al este del meridiano 180° y un polígono para la parte que se encuentra al oeste del meridiano:

```
MULTIPOLYGON(
  ((-180 30, -165 30, -165 40, -180 40, -180 30)),
  ((180 30, 180 40, 165 40, 165 30, 180 30)))
```

Tal como se muestra en la Figura 37 en la página 172, la representación terrestre redonda (Geodetic Data Management Feature) no requiere dicha división y puede utilizar un único polígono sin modificar:

```
POLYGON((165 30, -165 30, -165 40, 165 40, 165 30))
```

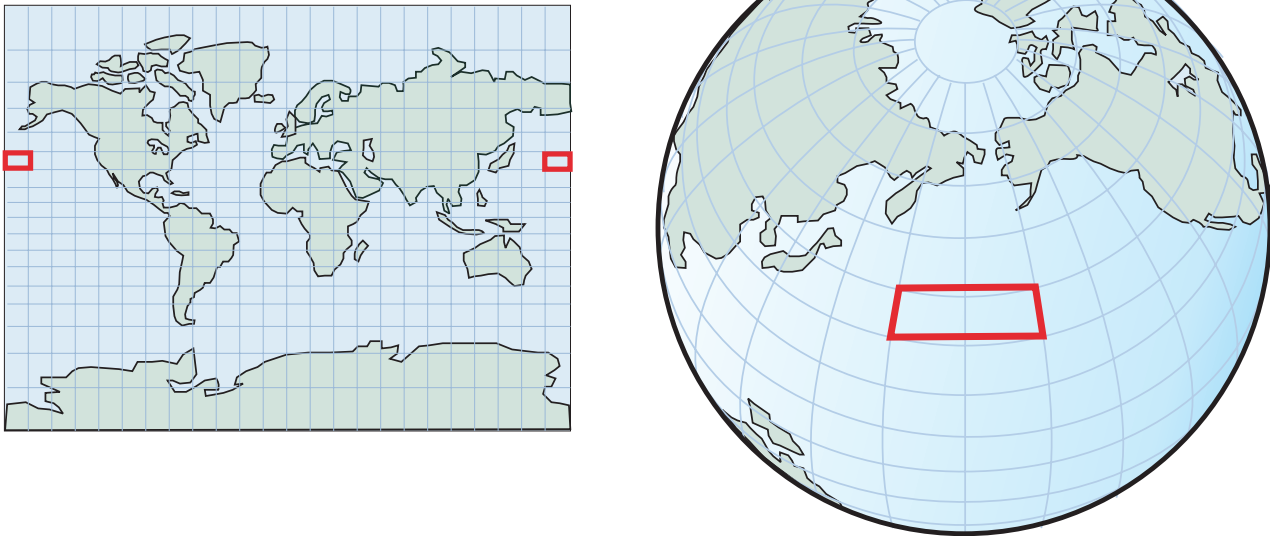


Figura 37. Polígonos que ocupan el meridiano 180°: creación de dos polígonos distintos

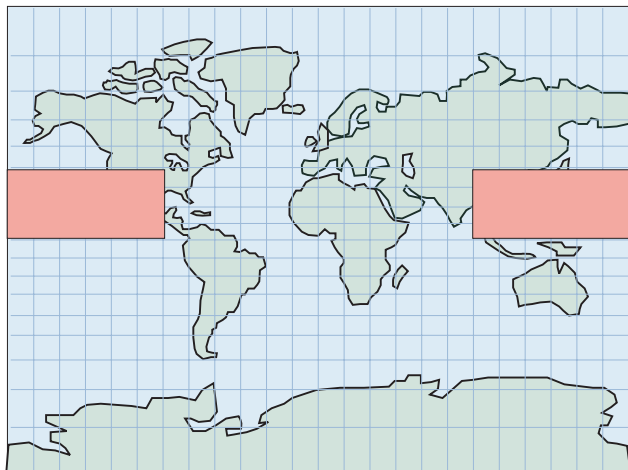
Si no ha creado dos polígonos separados al utilizar Spatial Extender, se reordenarían los vértices del polígono de modo que se defina un área distinta, tal como muestra la Figura 38 en la página 173. La parte superior de la Figura 38 en la página 173 muestra los vértices correctos de un polígono que ocupa el meridiano 180°:

```
POLYGON((90 0, -90 0, -90 40, 90 40, 90 0))
```

La parte inferior de la Figura 38 en la página 173 muestra los vértices reordenados que producen un polígono que ya no ocupa el meridiano 180°, sino que ahora ocupa el meridiano 0°.

```
POLYGON((-90 0, 90 0, 90 40, -90 40, -90 0))
```

Desea un polígono que abarque el meridiano 180:  
Polígono ((90 0, -90 0, -90 40, 90 40))



Pero Spatial Extender reorganiza los vértices y  
el polígono resultante define un área diferente:  
Polígono ((-90 0, 90 0, 90 40, -90 40))

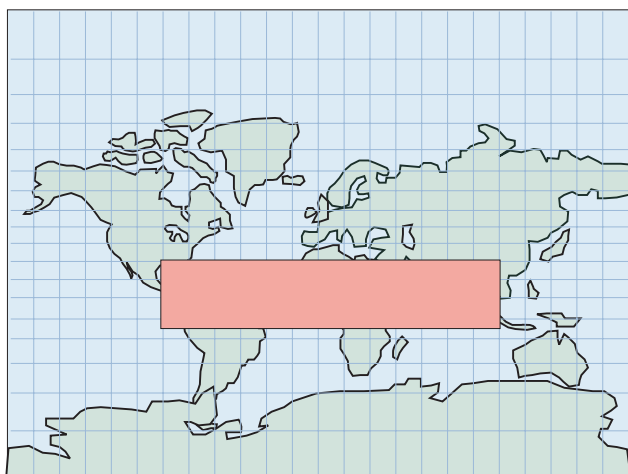


Figura 38. Polígonos que ocupan el meridiano 180°: vértices reordenados

El área definida sería el área complementaria de la tierra, y no el área de interés, tal como se muestra en la Figura 39 en la página 174. De forma similar al ejemplo del segmento lineal anterior, otra forma de manejar esta situación consiste en volver a proyectar los datos de modo que el meridiano 180° no esté en un borde.

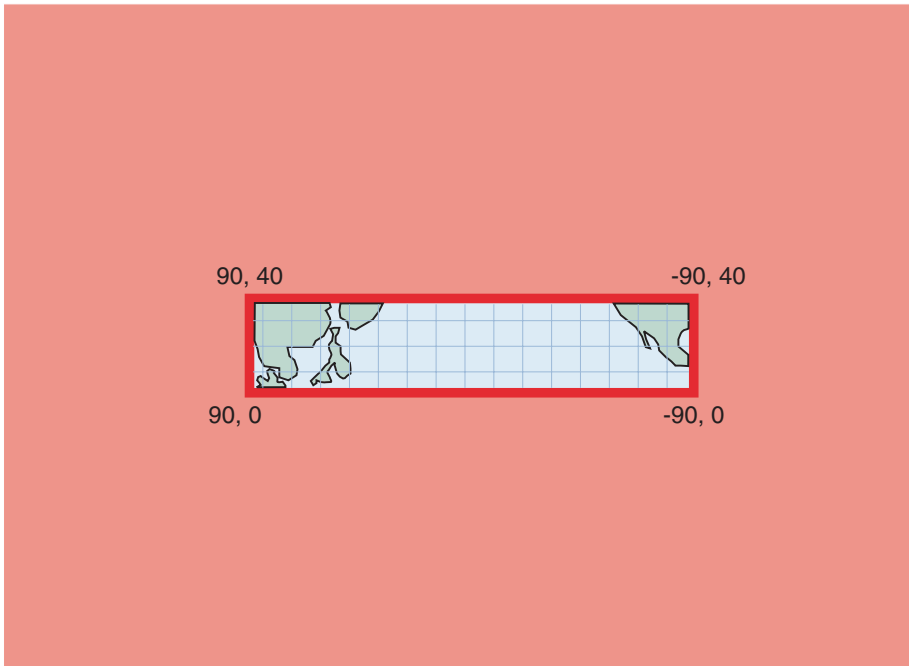


Figura 39. Polígonos que ocupan el meridiano 180°: área complementaria

## Polígonos que delimitan un polo

Como está trabajando directamente en el borde de la proyección de tierra plana con Spatial Extender, la distorsión del mapa de la superficie de la tierra le obliga a incorporar bordes y vértices adicionales para representar el polo dentro de un polígono.

En la Figura 40 en la página 175 se muestra cómo se podría trabajar con un polígono que delimitar el polo sur con Spatial Extender o con Geodetic Data Management Feature:

```
POLYGON((-180 -90, 180 -90, 180 -60, -180 -60, -180 -90))
```

La representación terrestre esférica(Geodetic Data Management Feature) muestra el polígono alrededor del polo sur como un círculo que sigue al paralelo sur -60°:

```
POLYGON((0 -60, -1 -60, -2 -60, ..., -179 -60, 180 -60, 179 -60, ..., 1 -60, 0 -60))
```

Una forma más adecuada de representar este círculo consiste en volver a proyectar los datos de forma que tanto el polo sur como el área circundante sean visibles en el mapa.



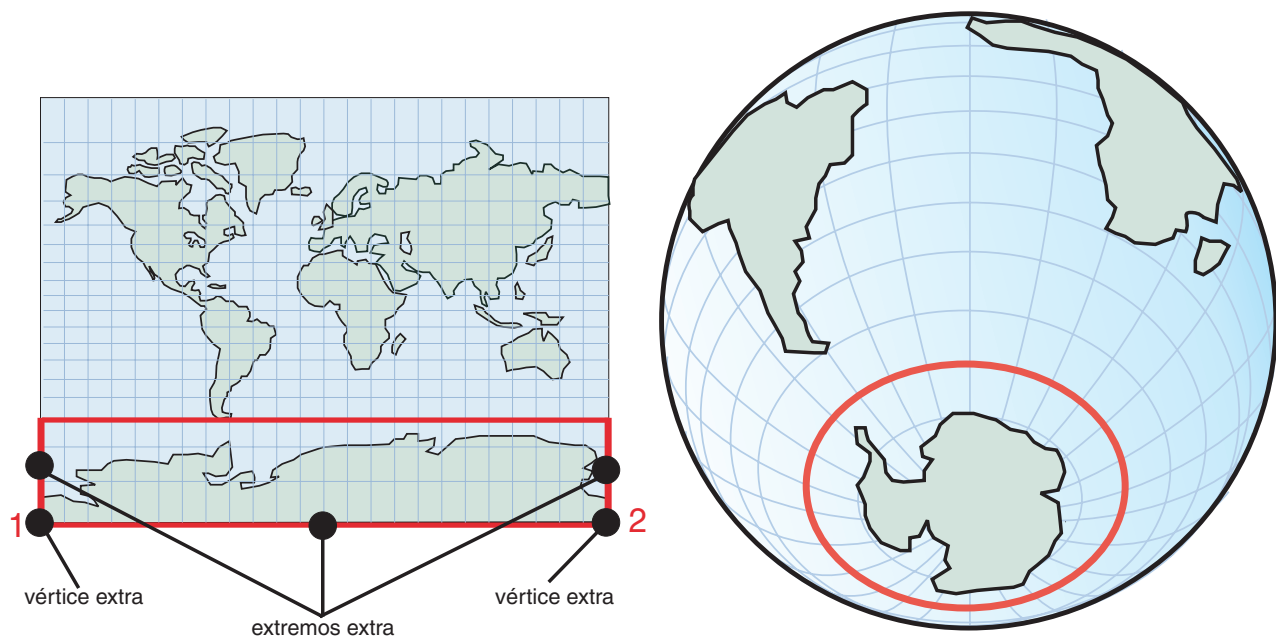


Figura 40. Polígonos que delimitan un polo

En los ejemplos anteriores puede obtener resultados precisos si elige un sistema de referencia espacial proyectado apropiado. Sin embargo, ninguna proyección puede resolverlos todos de forma simultánea. Por ejemplo, una proyección que no tenga el meridiano 180° en el borde coloca a este último en otro lugar y desplaza el área del problema.

## Polígonos que representan hemisferios, cinturones ecuatoriales y toda la tierra

Una representación terrestre redonda proporciona mejores resultados para cálculos de distancia y de área en un área grande de la superficie de la tierra.

Si necesita un polígono para representar áreas grandes de la superficie terrestre, como por ejemplo, uno de los hemisferios, los cinturones ecuatoriales o la propia tierra, deberá tener en cuenta las diferentes formas que Spatial Extender y Geodetic Data Management Feature manejan estos casos. En estas situaciones, una representación terrestre redonda obtiene resultados precisos para cálculos de distancia y área; por el contrario, una cuidadosa elección de la proyección no puede.

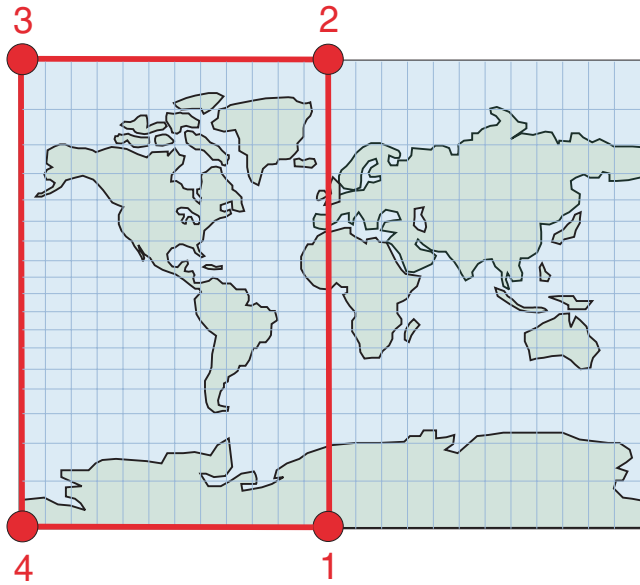
Por ejemplo, en la Figura 41 en la página 176 muestra que los polígonos que definen el hemisferio occidental en una representación terrestre plana (Geodetic Data Management Feature).

- En la representación terrestre plana en la parte superior de la Figura 41 en la página 176, las cuatro coordenadas representan el hemisferio occidental en un formato de texto convencional como 'POLYGON((0 -90, 0 90, -180 90, 180 -90, 0 -90))'.
- En la representación terrestre redonda, cuatro coordenadas representan el hemisferio occidental en formato de texto convencional como 'POLYGON((0 0, 0 90, 180 0, 0 -90, 0 0))'. Estas cuatro coordenadas definen un anillo alrededor de la tierra a lo largo del meridiano 0° y de su línea de las antípodas, el meridiano 180°.

Si especifica los mismos cuatro puntos en el orden inverso, está definiendo el hemisferio oriental:

- En una representación terrestre plana, el hemisferio oriental es 'POLYGON((0 -90, 180 -90, 180 90, 0 90, 0 -90))'.
- En una representación terrestre redonda, el hemisferio oriental es 'POLYGON((0 -90, 180 0, 0 90, 0 0, 0 -90))'.

Hemisferio occidental, representación plana  
Polígono ((0 -90, 0 90, -180 90, 180 -90, 0 -90))



Hemisferio occidental, representación terrestre  
Polígono ((0 0, 0 90, 180 0, 0 -90, 0 0))

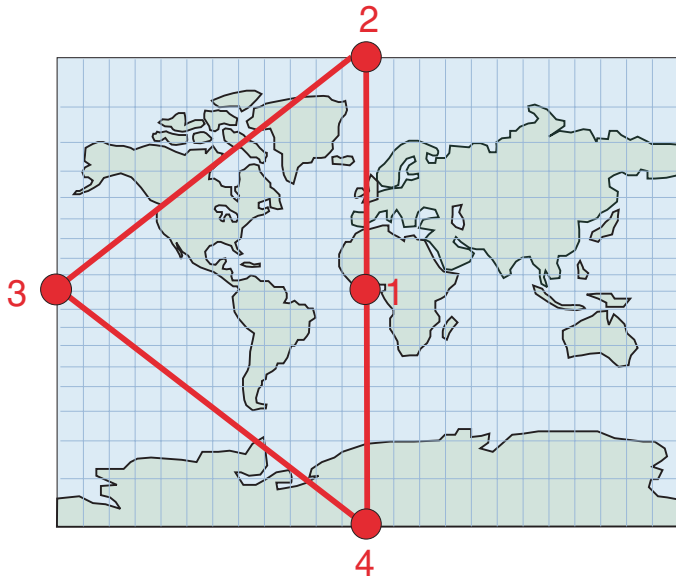


Figura 41. Polígonos que representan hemisferios

En la Figura 42 en la página 178 se muestran las coordenadas de polígonos que definen el cinturón ecuatorial en una representación terrestre redonda plana (Spatial Extender) y una representación terrestre redonda (Geodetic Data Management Feature).

- La parte superior de Figura 42 en la página 178 muestra la representación terrestre plana del cinturón ecuatorial con coordenadas en un formato de texto convencional como 'POLYGON((180 -60, 180 60, -180 60, -180 -60, 180 -60))'.
- En la representación terrestre redonda de la parte inferior de la Figura 42 en la página 178, se define el área de exclusión de dos anillos para representar el cinturón ecuatorial:

```
'MULTIPOLYGON(((0 60, -120 60, 120 60, 0 60)),  
((0 -60, 120 -60, -120 -60, 0 -60)))'
```

Sólo se muestran tres puntos de cada anillo para que quede más claro. En realidad, si desea que los anillos sigan de forma más cercana la línea de latitud 60 ó -60, deberá añadir más puntos intermedios. El primer anillo ((0 60, -120 60, 120 60, 0 60)) especifica los vértices en el orden que define el área sur de la línea de latitud 60. El segundo anillo ((0 -60, 120 -60, -120 -60, 0 -60)) especifica el área norte de la línea de latitud -60.

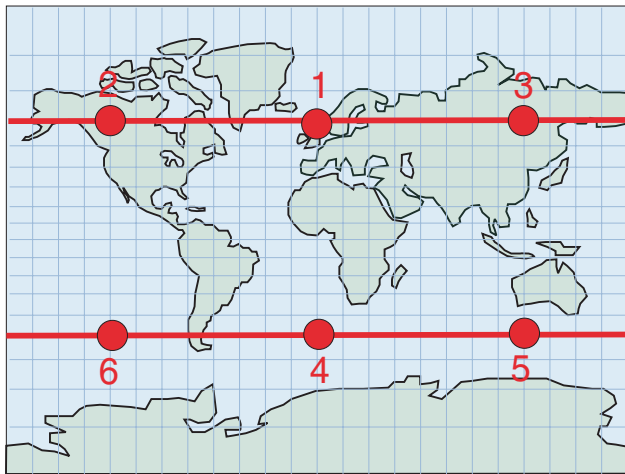
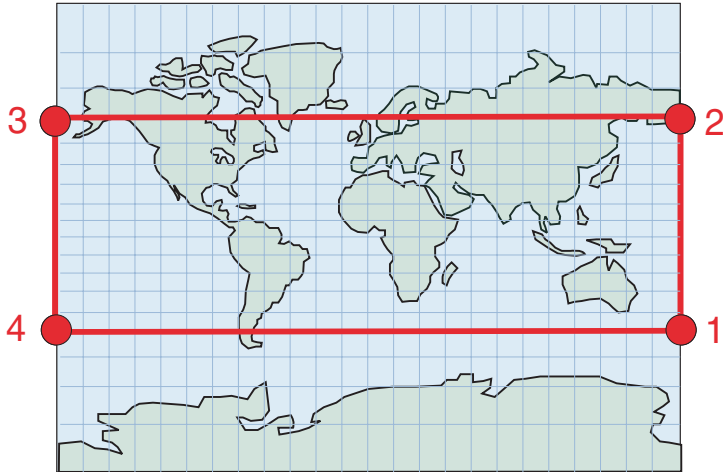
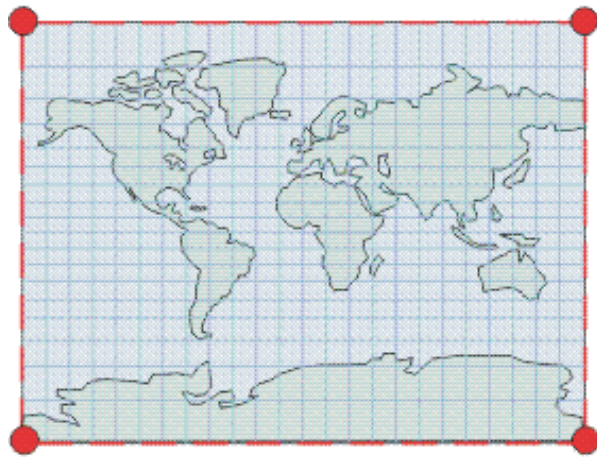
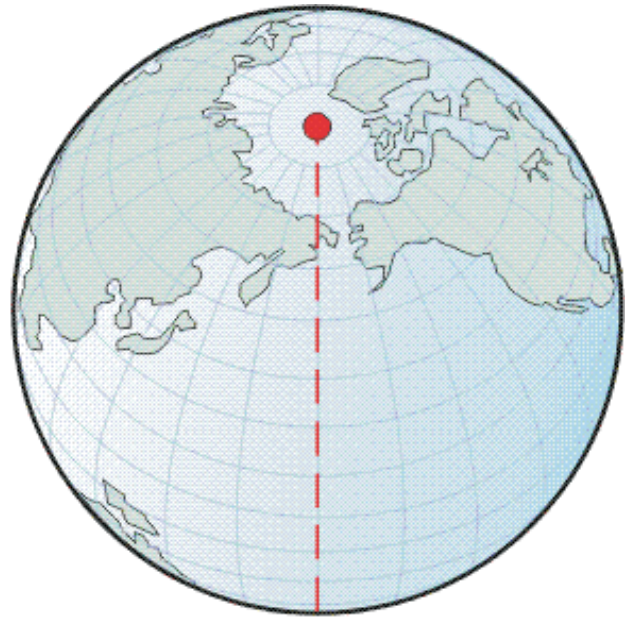


Figura 42. Polígonos que representan cinturones ecuatoriales

En la Figura 43 en la página 179 se muestran polígonos que definen la tierra al completo en una representación de tierra plana (Spatial Extender) y representación de tierra redonda (Geodetic Data Management Feature). Ambas representaciones representan toda la tierra con el mismo polígono en formato de texto convencional como 'POLYGON((-180 -90, 180 -90, 180 90, -180 90, -180 -90))'.



Representación plana



Representación elipsoide:  
Tal polígono no tiene límites no  
necesitando así una notación especial.

Figura 43. Polígonos que representan toda la tierra

## Funciones espaciales soportadas por DB2 Geodetic Data Management Feature

DB2 Spatial Extender se basa en la biblioteca de funciones proporcionada por ESRI y DB2 Geodetic Data Management Feature se basa en la biblioteca de funciones Hipparchus. Las diferencias de funcionamiento entre las bibliotecas ESRI e Hipparchus se traducen en diferencias menores en cuanto al comportamiento de algunas funciones. La tabla siguiente muestra funciones de Spatial Extender para las que Geodetic Data Management Feature tiene soporte e indica las diferencias de comportamiento. Para obtener información sobre el uso y la sintaxis de las funciones espaciales, consulte el apartado correspondiente sobre funciones espaciales.

Tabla 24. Soporte de funciones para Geodetic Data Management Feature

Función	¿Soportado por DB2 Geodetic Data Management Feature?	Diferencia de comportamiento para DB2 Geodetic Data Management Feature
EnvelopesIntersect	Sí	Ninguno
MBR Aggregate (Agregado MBR)	No	No aplicable
ST_AppendPoint	No	No aplicable
ST_Area	Sí	La unidad de medida por omisión es el metro.
ST_AsBinary	Sí	Ninguno
ST_AsGML	Sí	Ninguno

Tabla 24. Soporte de funciones para Geodetic Data Management Feature (continuación)

Función	¿Soportado por DB2 Geodetic Data Management Feature?	Diferencia de comportamiento para DB2 Geodetic Data Management Feature
ST_AsShape	Sí	Ninguno
ST_AsText	Sí	Ninguno
ST_Boundary	No	No aplicable
ST_Buffer	Sí	Soportada únicamente con puntos y multipuntos. La distancia puede ser un valor negativo. La unidad de medida por omisión es el metro.
ST_Centroid	No	No aplicable
ST_ChangePoint	No	No aplicable
ST_Contains	Sí	Ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.
ST_ConvexHull	No	No aplicable
ST_CoordDim	Sí	Ninguno
ST_Crosses	No	No aplicable
ST_Difference	Sí	No soportada con cadenas lineales y multilíneas. Ambas geometrías deben estar en el mismo SRS geodésico. La dimensión de la geometría devuelta es la misma que la de las geometrías de entrada.
ST_Dimension	Sí	Ninguno
ST_Disjoint	Sí	Ninguno
ST_Distance	Sí	Devuelve la <i>distancia geodésica</i> . Ambas geometrías deben estar en el mismo SRS geodésico. La unidad de medida por omisión es el metro.
ST_Edge_GC_USA	Sí	Ninguno
ST_Endpoint	Sí	Ninguno
ST_Envelope	Sí	La envoltura es un polígono que delimita el círculo delimitador mínimo (MBC) de la geometría.
ST_EnvIntersects	Sí	Ninguno
ST_EqualCoordsys	Sí	Ninguno
ST_Equals	No	No aplicable
ST_EqualSRS	Sí	Ninguno
ST_ExteriorRing	Sí	Ninguno
ST_FindMeasure o ST_LocateAlong	No	No aplicable
ST_Generalize	Sí	La unidad del umbral es el metro.
ST_GeomCollection	No	No aplicable
ST_GeomCollFromTxt	No	No aplicable
ST_GeomCollFromWKB	No	No aplicable
ST_Geometry	Sí	Ninguno
ST_GeometryN	Sí	Ninguno

Tabla 24. Soporte de funciones para Geodetic Data Management Feature (continuación)

Función	¿Soportado por DB2 Geodetic Data Management Feature?	Diferencia de comportamiento para DB2 Geodetic Data Management Feature
ST_GeometryType	Sí	Ninguno
ST_GeomFromText	Sí	Ninguno
ST_GeomFromWKB	Sí	Ninguno
ST_GetIndexParms	No	No aplicable
ST_InteriorRingN	Sí	Ninguno
ST_Intersection	Sí	La dimensión de la geometría devuelta es la de la entrada con la dimensión inferior, excepto cuando la dimensión de la intersección de dos cadenas sea 0.
ST_Intersects	Sí	Ambas geometrías deben estar en el mismo SRS geodésico.
ST_Is3d	Sí	Ninguno
ST_IsClosed	Sí	Ninguno
ST_IsEmpty	Sí	Ninguno
ST_IsMeasured	Sí	Ninguno
ST_IsRing	No	No aplicable
ST_IsSimple	No	No aplicable
ST_IsValid	Sí	Ninguno
ST_Length	Sí	La unidad de medida por omisión es el metro.
ST_LineFromText	Sí	Ninguno
ST_LineFromWKB	Sí	Ninguno
ST_LineString	Sí	Ninguno
ST_LineStringN	Sí	Ninguno
ST_M	Sí	Ninguno
ST_MaxM	Sí	Ninguno
ST_MaxX	Sí	Devuelve el valor X máximo del círculo delimitador mínimo (MBC). <b>Nota:</b> Si el MBC cruza la línea de datos, el valor de ST_MaxX es menor que el valor de ST_MinX. Si el MBC incluye el polo norte o el polo sur, ST_MinX es -180 y ST_MaxX es 180.
ST_MaxY	Sí	Devuelve el valor Y máximo del MBC. <b>Nota:</b> Si el MBC incluye el polo norte, el valor de ST_MaxY es 90.
ST_MaxZ	Sí	Ninguno
ST_MBR	Sí	MBR es una geometría que incluye el MBC de la geometría.
ST_MBRIntersects	Sí	Ninguno
ST_MeasureBetween o ST_LocateBetween	No	No aplicable
ST_MidPoint	Sí	Ninguno
ST_MinM	Sí	Ninguno

Tabla 24. Soporte de funciones para Geodetic Data Management Feature (continuación)

Función	¿Soportado por DB2 Geodetic Data Management Feature?	Diferencia de comportamiento para DB2 Geodetic Data Management Feature
ST_MinX	Sí	Devuelve el valor X mínimo del MBC. <b>Nota:</b> Si el MBC cruza la línea de datos, el valor de ST_MinX es mayor que el valor de ST_MaxX. Si el MBC incluye el polo norte o el polo sur, ST_MinX es -180 y ST_MaxX es 180.
ST_MinY	Sí	Devuelve el valor Y mínimo del MBC. <b>Nota:</b> Si el MBC incluye el polo sur, el valor ST_MinY es -90.
ST_MinZ	Sí	Ninguno
ST_MLineFromText	Sí	Ninguno
ST_MLineFromWKB	Sí	Ninguno
ST_MPointFromText	Sí	Ninguno
ST_MPointFromWKB	Sí	Ninguno
ST_MPolyFromText	Sí	Ninguno
ST_MPolyFromWKB	Sí	Ninguno
ST_MultiLineString	Sí	Ninguno
ST_MultiPoint	Sí	Ninguno
ST_MultiPolygon	Sí	Ninguno
ST_NumGeometries	Sí	Ninguno
ST_NumInteriorRing	Sí	Ninguno
ST_NumLineStrings	Sí	Ninguno
ST_NumPoints	Sí	Ninguno
ST_NumPolygons	Sí	Ninguno
ST_Overlaps	No	No aplicable
ST_Perimeter	Sí	La unidad de medida por omisión es el metro.
ST_PerpPoints	No	No aplicable
ST_Point	Sí	Ninguno
ST_PointFromText	Sí	Ninguno
ST_PointFromWKB	Sí	Ninguno
ST_PointN	Sí	Ninguno
ST_PolyFromText	Sí	Ninguno
ST_PolyFromWKB	Sí	Ninguno
ST_PointOnSurface	Sí	Ninguno
ST_Polygon	Sí	Ninguno
ST_PolygonN	Sí	Ninguno
ST_Relate	No	No aplicable
ST_RemovePoint	No	No aplicable
ST_SrsId o ST_SRID	Sí	Ninguno
ST_SrsName	Sí	Ninguno
ST_StartPoint	Sí	Ninguno



Tabla 24. Soporte de funciones para Geodetic Data Management Feature (continuación)

Función	¿Soportado por DB2 Geodetic Data Management Feature?	Diferencia de comportamiento para DB2 Geodetic Data Management Feature
ST_SymDifference	Sí	No soportada con cadenas lineales y multilíneas. La dimensión de la geometría devuelta es la misma que la de las geometrías de entrada. Ambas geometrías deben estar en el mismo SRS geodésico.
ST_ToGeomColl	No	No aplicable
ST_ToLineString	Sí	Ninguno
ST_ToMultiLine	Sí	Ninguno
ST_ToMultiPoint	Sí	Ninguno
ST_ToPoint	Sí	Ninguno
ST_ToPolygon	Sí	Ninguno
ST_Touches	No	No aplicable
ST_Transform	Sí	Ninguna. <b>Nota:</b> Las transformaciones de coordenadas se realizan punto por punto. Al transformar entre sistemas de coordenadas geodésicos y sistemas de coordenadas planas no proyectadas, preste atención a los polígonos y a las cadenas lineales que ocupan el meridiano 180°, o bien delimite uno o ambos polos. Debido a que Spatial Extender y Geodetic Data Management Feature tratan estos casos de forma distinta, es posible que las geometrías que son válidas en un sistema de coordenadas de tierra plana no sean válidas en un sistema de tierra redonda y viceversa. Para más información, consulte “Diferencias entre la utilización de representaciones terrestres y planas” en la página 169
ST_Union	Sí	Ambas geometrías deben estar en el mismo SRS geodésico.
ST_Within	Sí	Ambas geometrías deben estar en el mismo SRS geodésico.
ST_WKBToSQL	Sí	Ninguno
ST_WKTToSQL	Sí	Ninguno
ST_X	Sí	Ninguno
ST_Y	Sí	Ninguno
ST_Z	Sí	Ninguno
Union Aggregate (Agregado de unión)	No	No aplicable

## Procedimientos almacenados y vistas de catálogo de DB2 Geodetic Data Management Feature

DB2 Geodetic Data Management Feature da soporte a las mismas vistas de catálogo que DB2 Spatial Extender y a un subconjunto de los procedimientos almacenados espaciales.

Geodetic Data Management Feature no tiene soporte para los siguientes procedimientos almacenados:

- ST\_disable\_autogeocoding
- ST\_enable\_autogeocoding
- ST\_register\_geocoder
- ST\_remove\_geocoding\_setup
- ST\_run\_geocoding
- ST\_setup\_geocoding
- ST\_unregister\_geocoder

Geodetic Data Management Feature proporciona 318 sistemas de referencia espacial geodésicos predefinidos que aparecen en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS. Consulte la sección para obtener una lista completa.

---

## Sistemas de referencia soportados por DB2 Geodetic Data Management Feature

Tal como se describe, un *sistema de referencia* es un conjunto de valores que define la posición de un elipsoide relativo al centro de la tierra. Un sistema de referencia espacial (SRS) es un conjunto de parámetros que asocian un sistema de referencia con un elipsoide y se identifica con un identificador de sistema de referencia espacial (SRID). La Tabla 26 en la página 185 lista los sistemas de referencia predefinidos que proporciona DB2 Geodetic Data Management Feature. Los valores de desplazamiento y los factores de escala de todos los SRS geodésicos predefinidos son los mismos. La tabla siguiente muestra sus valores.

Tabla 25. Valores de desplazamiento y de escala de los SRS geodésicos predefinidos

Parámetro SRS	Valor
<i>xOffset</i>	-180
<i>yOffset</i>	-90
<i>zOffset</i>	-50000
<i>mOffset</i>	-1000
<i>xScale</i>	5965232
<i>yScale</i>	5965232
<i>zScale</i>	1000
<i>mScale</i>	1000

El parámetro *yScale* es siempre el mismo que *xScale*.

Puede elegir cualquier sistema de referencia que aparezca en la Tabla 26 en la página 185 para el sistema de referencia espacial. Elija el que se mejor se ajuste a sus datos. Por ejemplo, uno de los puntos de referencia más utilizados habitualmente, World Geodetic System 1984 (WGS 1984), toma el centro de la tierra como punto de origen y genera un mapa de todo el globo; se trata de un sistema de referencia centrado en la tierra. A diferencia de este, un sistema de referencia regional como North American 1927 genera un mapa de Norteamérica empezando desde un punto del suelo. Un sistema de referencia regional resulta preciso para la

región que pretende modelar, pero se necesita un sistema de referencia geodésico centrado en la tierra para manejar ubicaciones en todo el globo.

Tabla 26. SRID con sistema de referencia asociado y elipsoide

SRID	Nombre del sistema de referencia	Elipsoide de referencia
2000000000	WGS 1984	WGS 1984
2000000001	Abidjan 1987	Clarke 1880 (RGS)
2000000002	Accra	War Office
2000000003	Adindan	Clarke 1880 (RGS)
2000000004	Afgooye	Krasovsky 1940
2000000005	Agadez	Clarke 1880 (IGN)
2000000006	Australian Geodetic Datum 1966	Australian
2000000007	Australian Geodetic Datum 1984	Australian
2000000008	Ain el Abd 1970	International 1924
2000000009	Airy 1830	Airy 1830
2000000010	Airy Modified	Airy Modified
2000000011	Islas de Alaska	Clarke 1866
2000000012	Amersfoort	Bessel 1841
2000000013	Anguilla 1957	Clarke 1880 (RGS)
2000000014	Anna 1 Astro 1965	Australian
2000000015	Antigua Astro 1943	Clarke 1880 (RGS)
2000000016	Aratu	International 1924
2000000017	Arc 1950	Clarke 1880 (Arc)
2000000018	Arc 1960	Clarke 1880 (RGS)
2000000019	Ascension Island 1958	International 1924
2000000020	Assumed Geographic (NAD27 para archivos de formas sin PRJ)	Clarke 1866
2000000021	Astronomical Station 1952	International 1924
2000000022	ATF (Paris)	Plessis 1817
2000000023	Average Terrestrial System 1977	ATS 1977
2000000024	Australian National	Australian
2000000025	Ayabelle Lighthouse	Clarke 1880 (RGS)
2000000026	Bab South Astro (Bablethuap Is, República de Palau)	Clarke 1866
2000000027	Barbados 1938	Clarke 1880 (RGS)
2000000028	Batavia	Bessel 1841
2000000029	Batavia (Yakarta)	Bessel 1841
2000000030	Astro Beacon E 1945	International 1924
2000000031	Beduaram	Clarke 1880 (IGN)
2000000032	Beijing 1954	Krasovsky 1940
2000000033	Reseau National Belge 1950	International 1924
2000000034	Belge 1950 (Bruselas)	International 1924
2000000035	Reseau National Belge 1972	International 1924
2000000036	Bellevue (IGN)	International 1924
2000000037	Bermuda 1957	Clarke 1866
2000000038	Bern 1898	Bessel 1841
2000000039	Bern 1898 (Berna)	Bessel 1841
2000000040	Bern 1938	Bessel 1841
2000000041	Bessel 1841	Bessel 1841
2000000042	Bessel Modified	Bessel Modified
2000000043	Bessel Namibia	Bessel Namibia
2000000044	Bissau	International 1924
2000000045	Bogota	International 1924
2000000046	Bogota (Bogotá)	International 1924
2000000047	Bukit Rimpah	Bessel 1841

Tabla 26. SRID con sistema de referencia asociado y elipsoide (continuación)

SRID	Nombre del sistema de referencia	Elipsoide de referencia
2000000048	Camacupa	Clarke 1880 (RGS)
2000000049	Campo Inchauspe	International 1924
2000000050	Camp Area Astro	International 1924
2000000051	Canton Astro 1966	International 1924
2000000052	Cape	Clarke 1880 (Arc)
2000000053	Cape Canaveral	Clarke 1866
2000000054	Carthage	Clarke 1880 (IGN)
2000000055	Carthage (grados)	Clarke 1880 (IGN)
2000000056	Carthage (París)	Clarke 1880 (IGN)
2000000057	CH 1903	Bessel 1841
2000000058	CH 1903+	Bessel 1841
2000000059	Chatham Island Astro 1971	International 1924
2000000060	Chos Malal 1914	International 1924
2000000061	Swiss Terrestrial Ref. Frame 1995	GRS 1980
2000000062	Chua	International 1924
2000000063	Clarke 1858	Clarke 1858
2000000064	Clarke 1866	Clarke 1866
2000000065	Clarke 1866 (Michigan)	Clarke 1866 (Michigan)
2000000066	Clarke 1880	Clarke 1880
2000000067	Clarke 1880 (Arc)	Clarke 1880 (Arc)
2000000068	Clarke 1880 (Benoit)	Clarke 1880 (Benoit)
2000000069	Clarke 1880 (IGN)	Clarke 1880 (IGN)
2000000070	Clarke 1880 (RGS)	Clarke 1880 (RGS)
2000000071	Clarke 1880 (SGA)	Clarke 1880 (SGA)
2000000072	Conakry 1905	Clarke 1880 (IGN)
2000000073	Corrego Alegre	International 1924
2000000074	Cote d'Ivoire	Clarke 1880 (IGN)
2000000075	Dabola 1981	Clarke 1880 (RGS)
2000000076	Datum 73	International 1924
2000000077	Dealul Piscului 1933 (Rumanía)	International 1924
2000000078	Dealul Piscului 1970 (Rumanía)	Krasovsky 1940
2000000079	Deception Island	Clarke 1880 (RGS)
2000000080	Deir ez Zor	Clarke 1880 (IGN)
2000000081	Deutsche Hauptdreiecksnetz	Bessel 1841
2000000082	Dominica 1945	Clarke 1880 (RGS)
2000000083	DOS 1968	International 1924
2000000084	Astro DOS 71/4	International 1924
2000000085	Douala	Clarke 1880 (IGN)
2000000086	Easter Island 1967	International 1924
2000000087	European Datum 1950	International 1924
2000000088	European Datum 1950 (ED77)	International 1924
2000000089	European Datum 1987	International 1924
2000000090	Egypt 1907	Helmert 1906
2000000091	Estonia 1937	Bessel 1841
2000000092	Estonia 1992	GRS 1980
2000000093	European Terrestrial Ref. Frame 1989	WGS 1984
2000000094	European 1979	International 1924
2000000095	European Libyan Datum 1979	International 1924
2000000096	Everest 1830	Everest 1830
2000000097	Everest (Bangladesh)	Everest Adjustment 1937

Tabla 26. SRID con sistema de referencia asociado y elipsoide (continuación)

SRID	Nombre del sistema de referencia	Elipsoide de referencia
2000000098	Everest (Definition 1962)	Everest (Definition 1962)
2000000099	Everest (Definition 1967)	Everest (Definition 1967)
2000000100	Everest (Definition 1975)	Everest (Definition 1975)
2000000101	Everest (India y Nepal)	Everest (Definition 1962)
2000000102	Everest 1830 Modified	Everest 1830 Modified
2000000103	Everest Modified 1969	Everest Modified 1969
2000000104	Fahud	Clarke 1880 (RGS)
2000000105	Final Datum 1958	Clarke 1880 (RGS)
2000000106	Fischer 1960	Fischer 1960
2000000107	Fischer 1968	Fischer 1968
2000000108	Fischer Modified	Fischer Modified
2000000109	Fort Thomas 1955	Clarke 1880 (RGS)
2000000110	Gandajika 1970	International 1924
2000000111	Gan 1970	International 1924
2000000112	Garoua	Clarke 1880 (IGN)
2000000113	Geocentric Datum of Australia 1994	GRS 1980
2000000114	GEM 10C Gravity Potential Model	GEM 10C
2000000115	Greek Geodetic Ref. System 1987	GRS 1980
2000000116	Graciosa Base SW 1948	International 1924
2000000117	Greek	Bessel 1841
2000000118	Greek (Atenas)	Bessel 1841
2000000119	Grenada 1953	Clarke 1880 (RGS)
2000000120	GRS 1967	GRS 1967
2000000121	GRS 1980	GRS 1980
2000000122	Guam 1963	Clarke 1866
2000000123	Gunung Segara	Bessel 1841
2000000124	GUX 1 Astro	International 1924
2000000125	Guyane Francaise	International 1924
2000000126	Hanoi 1972	Krasovsky 1940
2000000127	Hartebeesthoek 1994	WGS 1984
2000000128	Helmert 1906	Helmert 1906
2000000129	Herat North	International 1924
2000000130	Hermannskogel	Bessel 1841
2000000131	Hito XVIII 1963	International 1924
2000000132	Hjorsey 1955	International 1924
2000000133	Hong Kong 1963	International 1924
2000000134	Hong Kong 1980	International 1924
2000000135	Hough 1960	Hough 1960
2000000136	Hungarian Datum 1972	GRS 1967
2000000137	Hu Tzu Shan	International 1924
2000000138	Indian 1954	Everest Adjustment 1937
2000000139	Indian 1960	Everest Adjustment 1937
2000000140	Indian 1975	Everest Adjustment 1937
2000000141	Indonesian National	Indonesian National
2000000142	Indonesian Datum 1974	Indonesian

Tabla 26. SRID con sistema de referencia asociado y elipsoide (continuación)

SRID	Nombre del sistema de referencia	Elipsoide de referencia
2000000143	International 1927	International 1924
2000000144	International 1967	International 1967
2000000145	IRENET95	GRS 1980
2000000146	Israel	GRS 1980
2000000147	ISTS 061 Astro 1968	International 1924
2000000148	ISTS 073 Astro 1969	International 1924
2000000149	Jamaica 1875	Clarke 1880
2000000150	Jamaica 1969	Clarke 1866
2000000151	Japan Geodetic Datum 2000	GRS 1980
2000000152	Johnston Island 1961	International 1924
2000000153	Kalianpur 1880	Everest 1830
2000000154	Kalianpur 1937	Everest Adjustment 1937
2000000155	Kalianpur 1962	Everest (Definition 1962)
2000000156	Kalianpur 1975	Everest (Definition 1975)
2000000157	Kandawala	Everest Adjustment 1937
2000000158	Kerguelen Island 1949	International 1924
2000000159	Kertau	Everest 1830 Modified
2000000160	Kartastokoordinaattijarjestelma	International 1924
2000000161	Kuwait Oil Company	Clarke 1880 (RGS)
2000000162	Korean Datum 1985	Bessel 1841
2000000163	Korean Datum 1995	WGS 1984
2000000164	Krasovsky 1940	Krasovsky 1940
2000000165	Kuwait Utility	GRS 1980
2000000166	Kusaie Astro 1951	International 1924
2000000167	Lake	International 1924
2000000168	La Canoa	International 1924
2000000169	L.C. 5 Astro 1961	Clarke 1866
2000000170	Leigon	Clarke 1880 (RGS)
2000000171	Liberia 1964	Clarke 1880 (RGS)
2000000172	Datum Lisboa Bessel	Bessel 1841
2000000173	Datum Lisboa Hayford	International 1924
2000000174	Lisbon (Lisboa)	International 1924
2000000175	Lisbon (Lisboa)	International 1924
2000000176	LKS 1994	GRS 1980
2000000177	Locodjo 1965	Clarke 1880 (RGS)
2000000178	Loma Quintana	International 1924
2000000179	Lome	Clarke 1880 (IGN)
2000000180	Luzon 1911	Clarke 1866
2000000181	Madrid 1870 (Merid. principal de Madrid)	Struve 1860
2000000182	Madzansua	Clarke 1866
2000000183	Mahe 1971	Clarke 1880 (RGS)
2000000184	Majuro (República de Marshall Is.)	Clarke 1866
2000000185	Makassar	Bessel 1841
2000000186	Makassar (Yakarta)	Bessel 1841
2000000187	Malongo 1987	International 1924
2000000188	Manoca	Clarke 1880 (RGS)
2000000189	Massawa	Bessel 1841
2000000190	Merchich	Clarke 1880 (IGN)

Tabla 26. SRID con sistema de referencia asociado y elipsoide (continuación)

SRID	Nombre del sistema de referencia	Elipsoide de referencia
2000000191	Merchich (grados)	Clarke 1880 (IGN)
2000000192	Militar-Geographische Institut	Bessel 1841
2000000193	MGI (Ferro)	Bessel 1841
2000000194	Mhast	International 1924
2000000195	Midway Astro 1961	International 1924
2000000196	Minna	Clarke 1880 (RGS)
2000000197	Monte Mario	International 1924
2000000198	Monte Mario (Rome)	International 1924
2000000199	Montserrat Astro 1958	Clarke 1880 (RGS)
2000000200	Mount Dillon	Clarke 1858
2000000201	Moznet	WGS 1984
2000000202	M'poraloko	Clarke 1880 (IGN)
2000000203	North American Datum 1927	Clarke 1866
2000000204	NAD 1927 CGQ77	Clarke 1866
2000000205	NAD 1927 (1976)	Clarke 1866
2000000206	North American Datum 1983	GRS 1980
2000000207	NAD 1983 (Sistema de ref. espaciales canadiense)	GRS 1980
2000000208	North American Datum 1983 (HARN)	GRS 1980
2000000209	NAD Michigan	Clarke 1866 (Michigan)
2000000210	Nahrwan 1967	Clarke 1880 (RGS)
2000000211	Naparima 1955	International 1924
2000000212	Naparima 1972	International 1924
2000000213	Nord de Guerre (París)	Plessis 1817
2000000214	National Geodetic Network (Kuwait)	WGS 1984
2000000215	NGO 1948	Bessel Modified
2000000216	NGO 1948 (Oslo)	Bessel Modified
2000000217	Nord Sahara 1959	Clarke 1880 (RGS)
2000000218	NSWC 9Z-2	NWL 9D
2000000219	Nouvelle Triangulation Francaise (grados)	Clarke 1880 (IGN)
2000000220	NTF (París) (grads)	Clarke 1880 (IGN)
2000000221	NWL 9D Transit Precise Ephemeris	NWL 9D
2000000222	New Zealand Geodetic Datum 1949	International 1924
2000000223	New Zealand Geodetic Datum 2000	GRS 1980
2000000224	Observatorio	Clarke 1866
2000000225	Observ. Meteorologico 1939	International 1924
2000000226	Old Hawaiian	Clarke 1866
2000000227	Oman	Clarke 1880 (RGS)
2000000228	OSGB 1936	Airy 1830
2000000229	OSGB 1970 (SN)	Airy 1830
2000000230	OSU 1986 Geoidal Model	OSU 86F
2000000231	OSU 1991 Geoidal Model	OSU 91A
2000000232	OS (SN) 1980	Airy 1830
2000000233	Padang 1884	Bessel 1841
2000000234	Padang 1884 (Yakarta)	Bessel 1841
2000000235	Palestine 1923	Clarke 1880 (Benoit)
2000000236	Pampa del Castillo	International 1924
2000000237	PDO Survey Datum 1993	Clarke 1880 (RGS)
2000000238	Pico de Las Nieves	International 1924
2000000239	Pitcairn Astro 1967	International 1924
2000000240	Plessis 1817	Plessis 1817

Tabla 26. SRID con sistema de referencia asociado y elipsoide (continuación)

SRID	Nombre del sistema de referencia	Elipsoide de referencia
2000000241	Pohnpei (Estados Fed. de Micronesia)	Clarke 1866
2000000242	Point 58	Clarke 1880 (RGS)
2000000243	Pointe Noire	Clarke 1880 (IGN)
2000000244	Porto Santo 1936	International 1924
2000000245	POSGAR	GRS 1980
2000000246	Provisional South Amer. Datum 1956	International 1924
2000000247	Puerto Rico	Clarke 1866
2000000248	Pulkovo 1942	Krasovsky 1940
2000000249	Pulkovo 1995	Krasovsky 1940
2000000250	Qatar 1974	International 1924
2000000251	Qatar 1948	Helmert 1906
2000000252	Qornoq	International 1924
2000000253	Rassadiran	International 1924
2000000254	REGVEN	GRS 1980
2000000255	Reunion	International 1924
2000000256	Reseau Geodesique Francais 1993	GRS 1980
2000000257	RT38	Bessel 1841
2000000258	RT38 (Estocolmo)	Bessel 1841
2000000259	RT 1990	Bessel 1841
2000000260	S-42 Hungary	Krasovsky 1940
2000000261	South American Datum 1969	GRS 1967 Truncated
2000000262	Samboja	Bessel 1841
2000000263	American Samoa 1962	Clarke 1866
2000000264	Santo DOS 1965	International 1924
2000000265	Sao Braz	International 1924
2000000266	Sapper Hill 1943	International 1924
2000000267	Schwarzeck	Bessel Namibia
2000000268	Segora	Bessel 1841
2000000269	Selvagem Grande 1938	International 1924
2000000270	Serindung	Bessel 1841
2000000271	Sierra Leone 1924	War Office
2000000272	Sierra Leone 1960	Clarke 1880 (RGS)
2000000273	Sierra Leone 1968	Clarke 1880 (RGS)
2000000274	SIRGAS	GRS 1980
2000000275	South Yemen	Krasovsky 1940
2000000276	Authalic sphere	Sphere
2000000277	Authalic sphere (ARC/INFO)	Sphere ARC INFO
2000000278	Struve 1860	Struve 1860
2000000279	St. George Island (Alaska)	Clarke 1866
2000000280	St. Kitts 1955	Clarke 1880 (RGS)
2000000281	St. Lawrence Island (Alaska)	Clarke 1866
2000000282	St. Lucia 1955	Clarke 1880 (RGS)
2000000283	St. Paul Island (Alaska)	Clarke 1866
2000000284	St. Vincent 1945	Clarke 1880 (RGS)
2000000285	Sudan	Clarke 1880 (IGN)
2000000286	South Asia Singapore	Fischer Modified
2000000287	S-JTSK	Bessel 1841
2000000288	S-JTSK (Ferro)	Bessel 1841
2000000289	Tananarive 1925	International 1924
2000000290	Tananarive 1925 (Paris)	International 1924
2000000291	Tern Island Astro 1961	International 1924
2000000292	Tete	Clarke 1866



Tabla 26. SRID con sistema de referencia asociado y elipsoide (continuación)

SRID	Nombre del sistema de referencia	Elipsoide de referencia
2000000293	Timbalai 1948	Everest (Definition 1967)
2000000294	TM65	Airy Modified
2000000295	TM75	Airy Modified
2000000296	Tokyo	Bessel 1841
2000000297	Trinidad 1903	Clarke 1858
2000000298	Tristan Astro 1968	International 1924
2000000299	Trucial Coast 1948	Helmert 1906
2000000300	Viti Levu 1916	Clarke 1880 (RGS)
2000000301	Voirol 1875	Clarke 1880 (IGN)
2000000302	Voirol 1875 (grados)	Clarke 1880 (IGN)
2000000303	Voirol 1875 (París)	Clarke 1880 (IGN)
2000000304	Voirol Unifie 1960	Clarke 1880 (RGS)
2000000305	Voirol Unifie 1960 (grados)	Clarke 1880 (RGS)
2000000306	Voirol Unifie 1960 (París)	Clarke 1880 (RGS)
2000000307	Wake-Eniwetok 1960	Hough 1960
2000000308	Wake Island Astro 1952	International 1924
2000000309	Walbeck	Walbeck
2000000310	War Office	War Office
2000000311	WGS 1966	WGS 1966
2000000312	WGS 1972	WGS 1972
2000000313	WGS 1972 Transit Broadcast Ephemeris	WGS 1972
2000000314	Yacare	International 1924
2000000315	Yemen Nat'l Geodetic Network 1996	WGS 1984
2000000316	Yoff	Clarke 1880 (IGN)
2000000317	Zanderij	International 1924

## Esferoides geodésicos

Un esferoide (también denominado elipsoide) es la parte de un sistema de coordenadas geográficas que define la forma de la superficie de la tierra en una ubicación determinada.

La definición de un sistema de coordenadas incluye la definición de un elipsoide en la definición de SPHEROID que forma parte de la definición de DATUM, tal como muestra el ejemplo siguiente:

```
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199432955]]
```

Puede utilizar la vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS para recuperar esta información. La columna **DEFINITION** de la vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS contiene los valores de las columnas **Nombre**, **Eje semimayor** y **Aplanamiento** de la tabla.



---

## Capítulo 20. Procedimientos almacenados

Esta sección proporciona información de consulta acerca de los procedimientos almacenados de DB2 Spatial Extender que puede utilizar para configurar DB2 Spatial Extender y crear proyectos que utilicen datos espaciales.

Cuando configure DB2 Spatial Extender o cree proyectos desde el Centro de control de DB2 o desde el procesador de línea de mandatos de DB2, puede invocar estos procedimientos almacenados implícitamente. Por ejemplo, cuando pulsa **Bien** en una ventana de DB2 Spatial Extender en el Centro de control de DB2, DB2 llama al procedimiento almacenado de DB2 Spatial Extender que está asociado con esa ventana.

De forma alternativa, puede invocar el procedimiento almacenado de DB2 Spatial Extender de forma explícita en un programa de aplicación.

Antes de invocar la mayoría de los procedimientos almacenados de DB2 Spatial Extender en una base de datos tendrá que realizar las tareas siguientes:

1. Asegúrese de que dispone de un espacio de tablas temporal del sistema con un tamaño de página de 8 KB o más y un tamaño mínimo de 500 páginas. Si no dispone de un espacio de tablas con estas características, consulte la sección sobre “creación de espacios de tablas temporales” en *Database Administration Concepts and Configuration Reference* para obtener más información sobre cómo crearlo. Es un requisito para poder ejecutar satisfactoriamente el procedimiento almacenado ST\_enable\_db o el mandato db2se enable\_db.
2. Para habilitar la base de datos para operaciones espaciales, invoque el procedimiento almacenado ST\_enable\_db stored procedure, de forma directa o utilizando el Centro de control de DB2. Consulte “ST\_enable\_db” en la página 216 para obtener más información.

Después de habilitar una base de datos para operaciones espaciales, puede invocar cualquiera de los procedimientos almacenados de DB2 Spatial Extender, bien implícita o explícitamente, en esa base de datos si está conectado a ella.

Este capítulo proporciona temas para todos los procedimientos almacenados de DB2 Spatial Extender, de la forma siguiente:

- “ST\_alter\_coordsys” en la página 194
- “ST\_alter\_srs” en la página 196
- “ST\_create\_coordsys” en la página 200
- “ST\_create\_srs” en la página 202
- “ST\_disable\_autogeocoding” en la página 208
- “ST\_disable\_db” en la página 210
- “ST\_drop\_coordsys” en la página 212
- “ST\_drop\_srs” en la página 213
- “ST\_enable\_autogeocoding” en la página 214
- “ST\_enable\_db” en la página 216
- “ST\_export\_shape” en la página 218
- “ST\_import\_shape” en la página 221
- “ST\_register\_geocoder” en la página 229

- “ST\_register\_spatial\_column” en la página 234
- “ST\_remove\_geocoding\_setup” en la página 235
- “ST\_run\_geocoding” en la página 237
- “ST\_setup\_geocoding” en la página 240
- “ST\_unregister\_geocoder” en la página 244
- “ST\_unregister\_spatial\_column” en la página 245

Las implementaciones de los procedimientos almacenados se archivan en la biblioteca db2gse en el servidor de DB2 Spatial Extender.

---

## ST\_alter\_coordsys

Utilice este procedimiento almacenado para actualizar una definición de sistema de coordenadas en la base de datos. Cuando se procesa este sistema almacenado, la información sobre el sistema de coordenadas se actualiza en la vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS.

**Atención:** Tenga cuidado con este procedimiento almacenado. Si utiliza este procedimiento almacenado para cambiar la definición del sistema de coordenadas y tiene datos espaciales existentes que están asociados con un sistema de referencia espacial que está basado en este sistema de coordenadas, es posible que sin darse cuenta modifique los datos espaciales. Si los datos espaciales resultan afectados, el usuario es responsable de asegurar que los datos espaciales modificados sigan siendo precisos y válidos.

### Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización DBADM.

### Sintaxis

```

▶▶ db2gse.ST_alter_coordsys( ( nombre_sistcoord, definición,
                             nulo )
▶ organización, id_sistcoord_organización, descripción )
  nulo          nulo          nulo

```

### Descripciones de parámetros

#### *nombre\_sistcoord*

Identifica de forma exclusiva el sistema de coordenadas. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *definición*

Define el sistema de coordenadas. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, la definición del sistema de coordenadas no cambia.

El tipo de datos de este parámetro es VARCHAR(2048).

#### *organización*

Designa la organización que definió el sistema de coordenadas y proporcionó la definición para él; por ejemplo, "European Petroleum Survey Group (EPSG)." Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si este parámetro tiene un valor nulo, la organización del sistema de coordenadas no cambia. Si este parámetro no es nulo, el parámetro *id\_sistcoord\_organización* no puede ser nulo; en este caso, la combinación de los parámetros *organización* y *id\_sistcoord\_organización* identifica de forma exclusiva el sistema de coordenadas.

El tipo de datos de este parámetro es VARCHAR(128).

#### *id\_sistcoord\_organización*

Especifica un identificador numérico que la entidad que se lista en el parámetro *organización* asigna a este sistema de coordenadas. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si este parámetro tiene un valor nulo, el parámetro *organización* debe tener también un valor nulo; en este caso, el identificador del sistema de coordenadas de la organización no cambia. Si este parámetro no tiene un valor nulo, el parámetro *organización* no puede tener un valor nulo; en este caso, la combinación de los parámetros *organización* y *id\_sistcoord\_organización* identifica de forma exclusiva el sistema de coordenadas.

El tipo de datos de este parámetro es INTEGER.

#### *descripción*

Describe el sistema de coordenadas explicando su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, la información de descripción sobre el sistema de coordenadas no cambia.

El tipo de datos de este parámetro es VARCHAR(256).

## **Parámetros de salida**

#### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

#### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_alter\_coordsys. Este ejemplo utiliza un mandato CALL de DB2 para actualizar un sistema de coordenadas denominado NORTH\_AMERICAN\_TEST. Este mandato CALL asigna un valor de 1002 al parámetro *id\_sistcoord*:

```
call db2gse.ST_alter_coordsys('NORTH_AMERICAN_TEST',NULL,NULL,1002,NULL,?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_alter\_srs

Utilice este procedimiento almacenado para actualizar una definición de sistema de referencia espacial en la base de datos. Cuando se procesa este procedimiento almacenado, la información acerca del sistema de referencia espacial se actualiza en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS.

Internamente, DB2 Spatial Extender almacena los valores de coordenadas como números enteros positivos. De esta manera, durante el cálculo, se puede reducir el impacto de errores de redondeo (que dependen en gran medida del valor real para las operaciones de coma flotante). El rendimiento de las operaciones espaciales también puede mejorar de forma significativa.

**Restricción:** No puede modificar un sistema de referencia si una columna espacial registrada utiliza dicho sistema de referencia espacial.

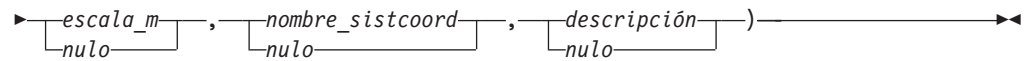
**Atención:** Tenga cuidado con este procedimiento almacenado. Si utiliza este procedimiento almacenado para cambiar los parámetros de desplazamiento, escala o *nombre\_sistcoord* del sistema de referencia espacial, y si tiene datos espaciales existentes que están asociados con el sistema de referencia espacial, es posible que sin darse cuenta modifique los datos espaciales. Si los datos espaciales resultan afectados, el usuario es responsable de asegurar que los datos espaciales modificados sigan siendo precisos y válidos.

## Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización DBADM.

## Sintaxis

```
▶▶ db2gse.ST_alter_srs ( nombre_srs , id_srs , desplazamiento_x  
                        [nulo] [nulo]  
▶ , escala_x , desplazamiento_y , escala_y ,  
                        [nulo] [nulo] [nulo]  
▶ desplazamiento_z , escala_z , desplazamiento_m ,  
                        [nulo] [nulo] [nulo]
```



## Descripciones de parámetros

### *nombre\_srs*

Identifica el sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *id\_srs*

Identifica de forma exclusiva el sistema de referencia espacial. Este identificador se utiliza como parámetro de entrada para varias funciones espaciales. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el identificador numérico del sistema de referencia espacial no cambia.

El tipo de datos de este parámetro es INTEGER.

### *desplazamiento\_x*

Especifica el desplazamiento de todas las coordenadas X de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El desplazamiento se resta antes de que se aplique el factor de escala *escala\_x* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. (WKT es texto convencional y WKB es binario convencional.)

El tipo de datos de este parámetro es DOUBLE.

### *escala\_x*

Especifica el factor de escala para todas las coordenadas X de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El factor de escala se aplica (multiplicación) después de restar el desplazamiento *x\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

### *desplazamiento\_y*

Especifica el desplazamiento para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El desplazamiento se resta antes de que se aplique el factor de escala *escala\_y* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_y*

Especifica el factor de escala para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El factor de escala se aplica (multiplicación) después de restar el desplazamiento *y\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Este factor de escala debe ser el mismo que *escala\_x*.

El tipo de datos de este parámetro es DOUBLE.

#### *desplazamiento\_z*

Especifica el desplazamiento para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El desplazamiento se resta antes de que se aplique el factor de escala *escala\_z* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_z*

Especifica el factor de escala para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El factor de escala se aplica (multiplicación) después de restar el desplazamiento *z\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

#### *desplazamiento\_m*

Especifica el desplazamiento para todas las coordenadas M de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El desplazamiento se resta antes de que se aplique el factor de escala *escala\_m* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_m*

Especifica el factor de escala para todas las coordenadas M de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.



El factor de escala se aplica (multiplicación) después de restar el desplazamiento *m\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

#### *nombre\_sistcoord*

Identifica de forma exclusiva el sistema de coordenadas en el que se basa este sistema de referencia espacial. El sistema de coordenadas debe estar listado en la vista ST\_COORDINATE\_SYSTEMS. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el sistema de coordenadas que se utiliza para este sistema de referencia espacial no cambia.

El valor de *nombre\_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *descripción*

Describe el sistema de referencia espacial explicando su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, la información de descripción sobre el sistema de referencia espacial no cambia.

El tipo de datos de este parámetro es VARCHAR(256).

## Parámetros de salida

#### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

#### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_alter\_srs. Este ejemplo utiliza un mandato CALL de DB2 para cambiar el valor del parámetro *descripción* de un sistema de referencia espacial denominado SRSDEMO:

```
call db2gse.ST_alter_srs('SRSDEMO',NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL,'SRS for GSE Demo Program: offices table',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_create\_coordsys

Utilice este procedimiento almacenado para almacenar información en la base de datos acerca del nuevo sistema de coordenadas. Cuando se procesa este procedimiento almacenado, la información acerca de este sistema de coordenadas se añade a la vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS.

### Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización DBADM.

### Sintaxis

```
db2gse.ST_create_coordsys(—nombre_sistcoord—,—definición—,——————>
—organización—,—id_sistcoord_organización—,—descripción—)————>
—nulo—,—nulo—,—nulo—)
```

### Descripciones de parámetros

#### *nombre\_sistcoord*

Identifica de forma exclusiva el sistema de coordenadas. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *definición*

Define el sistema de coordenadas. Debe especificar un valor que no sea nulo para este parámetro. El proveedor que suministra el sistema de coordenadas normalmente proporciona la información para este parámetro.

El tipo de datos de este parámetro es VARCHAR(2048).

#### *organización*

Designa la organización que definió el sistema de coordenadas y proporcionó la definición para él; por ejemplo, "European Petroleum Survey Group (EPSG)." Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si este parámetro tiene un valor nulo, el parámetro *id\_sistcoord\_organización* también debe tener un valor nulo. Si este parámetro no es nulo, el parámetro *id\_sistcoord\_organización* no puede ser nulo; en este caso, la combinación de los parámetros *organización* y *id\_sistcoord\_organización* identifica de forma exclusiva el sistema de coordenadas.

El tipo de datos de este parámetro es VARCHAR(128).

#### *id\_sistcoord\_organización*

Especifica un identificador numérico. La entidad que está especificada en el parámetro *organización* asigna este valor. Este valor no es necesariamente

exclusivo en todo el sistema de coordenadas. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si este parámetro tiene un valor nulo, el parámetro *organización* también debe tener un valor nulo. Si este parámetro no tiene un valor nulo, el parámetro *organización* no puede tener un valor nulo; en este caso, la combinación de los parámetros *organización* y *id\_sistcoord\_organización* identifica de forma exclusiva el sistema de coordenadas.

El tipo de datos de este parámetro es INTEGER.

#### *descripción*

Describe el sistema de coordenadas explicando su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se registra ninguna información de descripción sobre el sistema de coordenadas.

El tipo de datos de este parámetro es VARCHAR(256).

## Parámetros de salida

#### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

#### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_create\_coordsys. Este ejemplo utiliza un mandato CALL de DB2 para crear un sistema de coordenadas con los parámetros siguientes:

- parámetro *nombre\_sistcoord*: NORTH\_AMERICAN\_TEST
- parámetro *definición*:

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137.0,298.257222101]],
PRIMEM["Greenwich",0.0],
UNIT["Degree",0.0174532925199433]]
```
- parámetro *organización*: EPSG
- parámetro *id\_sistcoord\_organización*: 1001
- parámetro *descripción*: Test Coordinate Systems

```
call db2gse.ST_create_coordsys('NORTH AMERICAN TEST',
    'GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",
    SPHEROID["GRS_1980",6378137.0,298.257222101]],
    PRIMEM["Greenwich",0.0],UNIT["Degree",
    0.0174532925199433]]','EPSG',1001,'Test Coordinate Systems',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

## ST\_create\_srs

Utilice los procedimientos almacenados para crear un sistema de referencia espacial.

Un sistema de referencia espacial se define por el sistema de coordenadas, la precisión y las extensiones de las coordenadas que se representan en este sistema de referencia espacial. Las extensiones son los valores de las coordenadas mínimas y máximas para las coordenadas X, Y, Z y M.

Internamente, DB2 Spatial Extender almacena los valores de coordenadas como números enteros positivos. De esta manera, durante el cálculo, se puede reducir el impacto de errores de redondeo (que dependen en gran medida del valor real para las operaciones de coma flotante). El rendimiento de las operaciones espaciales también puede mejorar de forma significativa.

Este procedimiento almacenado tiene dos variaciones:

- La primera variación toma los factores de conversión (desplazamientos y factores de escala) como parámetros de entrada.
- La segunda variación toma las extensiones y la precisión como parámetros de entrada y calcula internamente los factores de conversión.

### Autorización

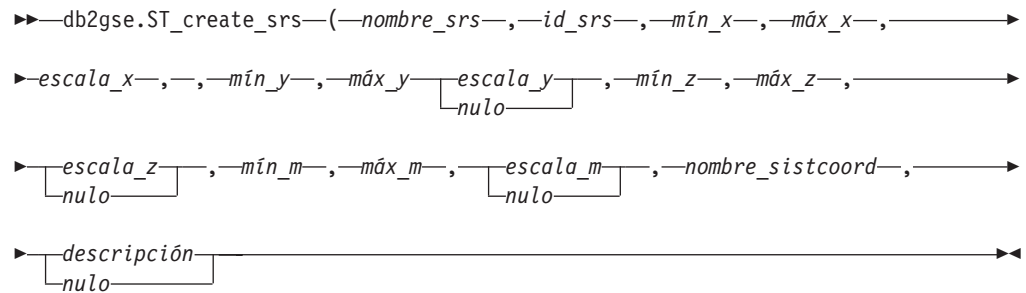
No se necesita ninguna.

### Sintaxis

#### Con factores de conversión (versión 1)

```
►► db2gse.ST_create_srs(—nombre_srs—, —id_srs—, —desplazamiento_x—, —
    └─nulo─┘
► escala_x—, —desplazamiento_y—, —escala_y—, —desplazamiento_z—
    └─nulo─┘ └─nulo─┘ └─nulo─┘
►, —escala_z—, —desplazamiento_m—, —escala_m—, —
    └─nulo─┘ └─nulo─┘ └─nulo─┘
► nombre_sistcoord—, —descripción—)
    └─nulo─┘
```

#### Con la máxima extensión posible (versión 2)



## Descripciones de parámetros

### Con factores de conversión (versión 1)

#### *nombre\_srs*

Identifica el sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *id\_srs*

Identifica de forma exclusiva el sistema de referencia espacial. Este identificador numérico se utiliza como parámetro de entrada para varias funciones espaciales. Debe especificar un valor que no sea nulo para este parámetro.

Para un sistema de referencia espacial geodésico, el valor *id\_srs* debe estar comprendido entre 2000000318 y 2000001000. DB2 Geodetic Data Management Feature proporciona sistemas de referencia espacial geodésicos predefinidos con valores *srs\_id* comprendidos entre 2000000000 y 2000000317.

El tipo de datos de este parámetro es INTEGER.

#### *desplazamiento\_x*

Especifica el desplazamiento de todas las coordenadas X de las geometrías que se representan en este sistema de referencia espacial. El desplazamiento se resta antes de que se aplique el factor de escala *escala\_x* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. (WKT es texto convencional, y WKB es binario convencional). Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero).

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_x*

Especifica el factor de escala para todas las coordenadas X de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *x\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Se especifica explícitamente el valor de *desplazamiento\_x* o se utiliza un valor por omisión de *desplazamiento\_x* de 0. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

#### *desplazamiento\_y*

Especifica el desplazamiento para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. El desplazamiento se resta antes de que se aplique el factor de escala *escala\_y* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero).

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_y*

Especifica el factor de escala para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *y\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Se especifica explícitamente un valor de *desplazamiento\_y* o se utiliza el valor por omisión de *desplazamiento\_y* de 0. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará el valor del parámetro *escala\_x*. Si especifica un valor distinto de nulo para este parámetro, el valor que especifique debe coincidir con el valor del parámetro *escala\_x*.

El tipo de datos de este parámetro es DOUBLE.

#### *desplazamiento\_z*

Especifica el desplazamiento para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. El desplazamiento se resta antes de que se aplique el factor de escala *escala\_z* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero).

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_z*

Especifica el factor de escala para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *z\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Se especifica explícitamente el valor de *desplazamiento\_z* o se utiliza un valor por omisión de *desplazamiento\_z* de 0. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 1.

El tipo de datos de este parámetro es DOUBLE.

#### *desplazamiento\_m*

Especifica el desplazamiento para todas las coordenadas M de las geometrías que se representan en este sistema de referencia espacial. El desplazamiento se resta antes de que se aplique el factor de escala *escala\_m* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero).

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_m*

Especifica el factor de escala para todas las coordenadas M de las geometrías

que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *m\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Se especifica explícitamente el valor de *desplazamiento\_m* o se utiliza un valor por omisión de *desplazamiento\_m* de 0. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 1.

El tipo de datos de este parámetro es DOUBLE.

#### *nombre\_sistcoord*

Identifica de forma exclusiva el sistema de coordenadas en el que se basa este sistema de referencia espacial. El sistema de coordenadas debe estar listado en la vista ST\_COORDINATE\_SYSTEMS. Debe proporcionar un valor que no sea nulo para este parámetro.

El valor de *nombre\_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *descripción*

Describe el sistema de referencia espacial explicando el propósito de su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se registrará ninguna información de descripción.

El tipo de datos de este parámetro es VARCHAR(256).

### **Con la máxima extensión posible (versión 2)**

#### *nombre\_srs*

Identifica el sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *id\_srs*

Identifica de forma exclusiva el sistema de referencia espacial. Este identificador numérico se utiliza como parámetro de entrada para varias funciones espaciales. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es INTEGER.

#### *mín\_x*

Especifica el valor mínimo posible de coordenada X para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

#### *máx\_x*

Especifica el valor máximo posible de coordenada X para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

En función del valor de *escala\_x*, el valor que se muestra en la vista DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS puede ser mayor que el valor especificado aquí. El valor de la vista es correcto.

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_x*

Especifica el factor de escala para todas las coordenadas X de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *x\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. El cálculo del desplazamiento *desplazamiento\_x* se basa en el valor *x\_min*. Debe proporcionar un valor que no sea nulo para este parámetro.

Si se especifican los parámetros *escala\_x* y *escala\_y*, los valores deben coincidir.

El tipo de datos de este parámetro es DOUBLE.

#### *mín\_y*

Especifica el valor mínimo posible de coordenada Y para todas las geometrías que utilizan este sistema de referencia espacial. Debe proporcionar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

#### *máx\_y*

Especifica el valor máximo posible de coordenada Y para todas las geometrías que utilizan este sistema de referencia espacial. Debe proporcionar un valor que no sea nulo para este parámetro.

En función del valor de *escala\_y*, el valor que se muestra en la vista DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS puede ser mayor que el valor especificado aquí. El valor de la vista es correcto.

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_y*

Especifica el factor de escala para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *y\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. El cálculo del desplazamiento *desplazamiento\_y* se basa en el valor *y\_min*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará el valor del parámetro *escala\_x*. Si se especifican los parámetros *escala\_y* y *escala\_x*, los valores deben coincidir.

El tipo de datos de este parámetro es DOUBLE.

#### *mín\_z*

Especifica el valor mínimo posible de coordenadas Z para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

#### *máx\_z*

Especifica el valor máximo posible de coordenada Z para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.



En función del valor de *escala\_z*, el valor que se muestra en la vista DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS puede ser mayor que el valor especificado aquí. El valor de la vista es correcto.

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_z*

Especifica el factor de escala para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *z\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. El cálculo del desplazamiento *desplazamiento\_z* se basa en el valor *mín\_z*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 1.

El tipo de datos de este parámetro es DOUBLE.

#### *mín\_m*

Especifica el valor mínimo posible de coordenada M para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

#### *máx\_m*

Especifica el valor máximo posible de coordenada M para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

En función del valor de *escala\_m*, el valor que se muestra en la vista DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS puede ser mayor que el valor especificado aquí. El valor de la vista es correcto.

El tipo de datos de este parámetro es DOUBLE.

#### *escala\_m*

Especifica el factor de escala para todas las coordenadas M de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *m\_offset* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. El cálculo del desplazamiento *desplazamiento\_m* se basa en el valor *mín\_m*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 1.

El tipo de datos de este parámetro es DOUBLE.

#### *nombre\_sistcoord*

Identifica de forma exclusiva el sistema de coordenadas en el que se basa este sistema de referencia espacial. El sistema de coordenadas debe estar listado en la vista ST\_COORDINATE\_SYSTEMS. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *descripción*

Describe el sistema de referencia espacial explicando el propósito de su

aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se registrará ninguna información de descripción.

El tipo de datos de este parámetro es VARCHAR(256).

## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_create\_srs. Este ejemplo utiliza un mandato CALL de DB2 para crear un sistema de referencia espacial denominado SRSDEMO con los valores de parámetros siguientes:

- *id\_srs*: 1000000
- *desplazamiento\_x*: -180
- *escala\_x*: 1000000
- *desplazamiento\_y*: -90
- *escala\_y*: 1000000

```
call db2gse.ST_create_srs('SRSDEMO',1000000,  
                        -180,1000000, -90, 1000000,  
                        0, 1, 0, 1,'NORTH_AMERICAN',  
                        'SRS for GSE Demo Program: customer table',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_disable\_autogeocoding

Utilice este procedimiento almacenado para especificar que DB2 Spatial Extender deje de sincronizar una columna geocodificada con su columna o columnas de geocodificación asociadas.

Una *columna de geocodificación* se utiliza como dato de entrada en el geocodificador.

## Autorización

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las autorizaciones o uno de los privilegios siguientes:

- Autorización DBADM y DATAACCESS sobre la base de datos que contiene la tabla en la que están definidos los desencadenantes que se van a descartar
- Privilegio CONTROL sobre esta tabla
- Privilegios ALTER y UPDATE sobre esta tabla

**Nota:** Para los privilegios CONTROL y ALTER, debe tener autorización DROPIN sobre el esquema DB2GSE.

## Sintaxis

```
►► db2gse.ST_disable_autogeocoding—(—esquema_tabla—, —nombre_tabla—, —  
                                          —nulo—)  
►—nombre_columna—)◄◄
```

## Descripciones de parámetros

### *esquema\_tabla*

Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre\_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_tabla*

Especifica el nombre no calificado de la tabla en la que están definidos los desencadenantes que desea descartar. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_columna*

Especifica la columna geocodificada que es mantenida por los desencadenantes que desea descartar. Debe especificar un valor que no sea nulo para este parámetro.

El valor *nombre\_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_disable\_autogeocoding. Este ejemplo utiliza un mandato CALL de DB2 para inhabilitar la geocodificación en la columna LOCATION de la tabla denominada CUSTOMERS:

```
call db2gse.ST_disable_autogeocoding(NULL, 'CUSTOMERS', 'LOCATION', ?, ?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_disable\_db

Utilice este procedimiento almacenado para eliminar recursos que permitan a DB2 Spatial Extender almacenar y dar soporte a datos espaciales.

Este procedimiento almacenado le ayuda a solucionar problemas o temas que surjan tras habilitar la base de datos para operaciones espaciales. Por ejemplo, puede habilitar una base de datos para operaciones espaciales y a continuación decidir utilizar en su lugar otra base de datos con DB2 Spatial Extender. Si no ha definido ninguna columna espacial ni ha importado datos espaciales, puede invocar a este procedimiento almacenado para eliminar todos los recursos espaciales de la primera base de datos. Debido a esta interdependencia entre las columnas espaciales y las definiciones de tipos, no puede descartar las definiciones de tipos cuando existan columnas de estos tipos. Si ya ha definido las columnas espaciales pero todavía desea inhabilitar una base de datos para operaciones espaciales, debe especificar un valor distinto de 0 (cero) para el parámetro *forzar* para eliminar todos los recursos espaciales de la base de datos que no tienen otras dependencias en ellos.

## Autorización

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización DBADM sobre la base de datos de la que se van a eliminar los recursos de DB2 Spatial Extender.

## Sintaxis

► db2gse.ST\_disable\_db(—[forzar]—)  
                                  └─nulo─┘

## Descripciones de parámetros

### *forzar*

Especifica que desea inhabilitar una base de datos para operaciones espaciales, aunque puede tener objetos de base de datos que sean dependientes de los tipos espaciales o de las funciones espaciales. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si especifica un valor distinto de 0 (cero) o nulo para el parámetro *force*, la base de datos se inhabilita y se eliminan todos los recursos de DB2 Spatial Extender (si es posible). Si especifica 0 (cero) o nulo, la base de datos no se inhabilita si los objetos de base de datos son dependientes de tipos espaciales o de funciones espaciales. Los objetos de base de datos que pueden tener este tipo de dependencias incluyen tablas, vistas, restricciones, desencadenantes, columnas generadas, métodos, funciones, procedimientos y otros tipos de datos (subtipos o tipos estructurados con un atributo espacial).

El tipo de datos de este parámetro es SMALLINT.

## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_disable\_db. Este ejemplo utiliza un mandato CALL de DB2 para inhabilitar la base de datos para operaciones espaciales, con un valor del parámetro *force* de 1:

```
call db2gse.ST_disable_db(1,?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_drop\_coordsys

Utilice este procedimiento almacenado para suprimir información acerca de un sistema de coordenadas de la base de datos. Cuando se procesa este procedimiento almacenado, la información sobre el sistema de coordenadas se elimina de la vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS.

### Restricción:

No puede descartar un sistema de coordenadas en el que se basa un sistema de referencia espacial.

### Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización DBADM.

### Sintaxis

►►—db2gse.ST\_drop\_coordsys—(—*nombre\_sistcoord*—)—————►►

### Descripciones de parámetros

#### *nombre\_sistcoord*

Identifica de forma exclusiva el sistema de coordenadas. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### Parámetros de salida

#### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

#### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

### Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_drop\_coordsys. Este ejemplo utiliza

un mandato CALL de DB2 para suprimir de la base de datos un sistema de coordenadas denominado NORTH\_AMERICAN\_TEST:

```
call db2gse.ST_drop_coordsys('NORTH_AMERICAN_TEST',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_drop\_srs

Utilice este procedimiento almacenado para descartar un sistema de referencia espacial.

Cuando se procesa este procedimiento almacenado, la información sobre el sistema de referencia espacial se elimina de la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS.

**Restricción:** No puede descartar un sistema de referencia espacial si una columna espacial que utiliza dicho sistema de referencia espacial está registrado.

### Importante:

Tenga cuidado al utilizar este procedimiento almacenado. Si utiliza este procedimiento almacenado para descartar un sistema de referencia espacial y si algunos datos espaciales están asociados con este sistema de referencia espacial, ya no puede realizar operaciones espaciales sobre los datos espaciales.

### Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización DBADM.

### Sintaxis

```
►►—db2gse.ST_drop_srs—(—nombre_srs—)—————►►
```

### Descripciones de parámetros

#### *nombre\_srs*

Identifica el sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### Parámetros de salida

#### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

*texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_drop\_srs. Este ejemplo utiliza un mandato CALL de DB2 para suprimir un sistema de referencia espacial denominado SRSDEMO:

```
call db2gse.ST_drop_srs('SRSDEMO',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_enable\_autogeocoding

Utilice este procedimiento almacenado para especificar que DB2 Spatial Extender sincronice una columna geocodificada con su columna o sus columnas de geocodificación asociadas.

Una *columna de geocodificación* se utiliza como dato de entrada en el geocodificador. Cada vez que se insertan o actualizan valores en la columna o columnas de geocodificación, se activan desencadenantes. Estos desencadenantes invocan al geocodificador asociado para geocodificar los valores insertados o actualizados y para colocar los datos resultantes en la columna geocodificada.

**Restricción:** Puede habilitar la geocodificación automática sólo en tablas para las que se pueden crear desencadenantes de inserción (INSERT) y actualización (UPDATE). En consecuencia, no puede habilitar la geocodificación automática en vistas ni apodos.

**Requisito necesario:** Antes de habilitar la geocodificación automática, debe realizar el paso de definición de la geocodificación invocando al procedimiento ST\_setup\_geocoding. El paso de definición de la geocodificación especifica los valores del geocodificador y del parámetro de geocodificación. Además, identifica las columnas de geocodificación que se van a sincronizar con las columnas geocodificadas.

## Autorización

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las autorizaciones o uno de los privilegios siguientes:

- Autorización DBADM sobre la base de datos que contiene la tabla en la que están definidos los desencadenantes que ha creado este procedimiento almacenado
- Privilegio CONTROL sobre la tabla
- Privilegio ALTER sobre la tabla



Si el ID de autorización de la sentencia no tiene autorización DBADM, los privilegios que tenga el ID de autorización de la sentencia (sin tener en cuenta privilegios PUBLIC o de grupo) deben incluir todos los privilegios siguientes mientras exista el desencadenante:

- Privilegio SELECT sobre la tabla en la que está habilitada la geocodificación automática o autorización DATAACCESS
- Privilegios necesarios para evaluar las expresiones de SQL que se especifican para los parámetros en la configuración de geocodificación

## Sintaxis

```
► db2gse.ST_enable_autogeocoding—(—esquema_tabla—,—nombre_tabla—,—  
                                          nulo—)———  
► nombre_columna—)———
```

## Descripciones de parámetros

### *esquema\_tabla*

Identifica el esquema al que pertenece la tabla que está especificada en el parámetro *nombre\_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla.

El valor de *esquema\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_tabla*

Especifica el nombre no calificado de la tabla que contiene la columna en la que se insertarán o se actualizarán los datos geocodificados. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_columna*

Identifica la columna en la que se insertarán o actualizarán los datos geocodificados. Esta columna se denomina columna geocodificada. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de

finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

*texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_enable\_autogeocoding. Este ejemplo utiliza un mandato CALL de DB2 para habilitar la geocodificación en la columna LOCATION de la tabla denominada CUSTOMERS:

```
call db2gse.ST_enable_autogeocoding(NULL,'CUSTOMERS','LOCATION',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_enable\_db

Utilice este procedimiento almacenado para proporcionar a la base de datos los recursos que necesita para almacenar datos espaciales y para dar soporte a operaciones espaciales. Estos recursos incluyen tipos de datos espaciales, tipos de índices espaciales, vistas de catálogo, funciones proporcionadas y otros procedimientos almacenados.

Este procedimiento almacenado sustituye a db2gse.gse\_enable\_db.

## Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización DBADM sobre la base de datos que se está habilitando.

## Sintaxis

```
▶▶ db2gse.ST_enable_db ( ( parámetros_creación_tabla ) ) ▶▶  
└─ nulo ─┘
```

## Descripciones de parámetros

*parámetros\_creación\_tabla*

Especifica las opciones que se van a añadir a las sentencias CREATE TABLE para las tablas de catálogo de DB2 Spatial Extender. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se añade ninguna opción a las sentencias CREATE TABLE.

Para especificar estas opciones, utilice la sintaxis de la sentencia DB2 CREATE TABLE. Por ejemplo, para especificar un espacio de tabla en el que se crearán las tablas, utilice:

```
IN tsName INDEX IN indexTsName
```

El tipo de datos de este parámetro es VARCHAR(32K).

## Parámetros de salida

*código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

*texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

El siguiente ejemplo muestra cómo utilizar la CLI (Call Level Interface) para invocar al procedimiento almacenado ST\_enable\_db:

```
SQLHANDLE henv;
SQLHANDLE hdbc;
SQLHANDLE hstmt;
SQLCHAR uid[MAX_UID_LENGTH + 1];
SQLCHAR pwd[MAX_PWD_LENGTH + 1];
SQLINTEGER ind[3];
SQLINTEGER msg_code = 0;
char msg_text[1024] = "";
SQLRETURN rc;
char *table_creation_parameters = NULL;

/* Asignar descriptor de entorno */
rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);

/* Asignar descriptor de base de datos */
rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);

/* Establecer una conexión con la base de datos "testdb" */
rc = SQLConnect(hdbc, (SQLCHAR *)"testdb", SQL_NTS, (SQLCHAR *)uid,
                SQL_NTS, (SQLCHAR *)pwd, SQL_NTS);

/* Asignar descriptor de sentencia */
rc = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt) ;

/* Asociar sentencia de SQL para llamar al procedimiento almacenado ST_enable_db con el descriptor de sentencia y
/* enviar la sentencia a DBMS para que esté preparado.
*/
rc = SQLPrepare(hstmt, "call db2gse!ST_enable_db(?,?,?)", SQL_NTS);
```

```

/* Vincular el primer marcador de parámetro de la sentencia de */
/* llamada de SQL, el parámetro de entrada de los parámetros de */
/* creación de tablas, con la variable table_creation_parameters. */
ind[0] = SQL_NULL_DATA;
rc = SQLBindParameter(hstmt, 1, SQL_PARAM_OUTPUT, SQL_C_CHAR,
    SQL_VARCHAR, 255, 0, table_creation_parameters, 256, &ind[0]);

/* Vincular el segundo marcador de parámetro de la sentencia de */
/* de llamada de SQL, el parámetro de salida del código de mensaje */
/* devuelto, con la variable msg_code. */
ind[1] = 0;
rc = SQLBindParameter(hstmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
    SQL_INTEGER, 0, 0, &msg_code, 4, &ind[1]);

/* Vincular el tercer marcador de parámetro de la sentencia */
/* SQL, el parámetro de salida */
/* devuelto, con la variable msg_text. */
ind[2] = 0;
rc = SQLBindParameter(hstmt, 3, SQL_PARAM_OUTPUT, SQL_C_CHAR,
    SQL_VARCHAR, (sizeof(msg_text)-1), 0, msg_text,
    sizeof(msg_text), &ind[2]);
rc = SQLExecute(hstmt);

```

---

## ST\_export\_shape

Utilice este procedimiento almacenado para exportar una columna espacial y su tabla asociada a un archivo de formas.

### Autorización

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener los privilegios necesarios para ejecutar satisfactoriamente la sentencia SELECT desde la que se van a exportar los datos.

El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para crear los archivos de formas y escribir en los mismos.

### Sintaxis

```

▶▶ db2gse.ST_export_shape (—nombre_archivo—, —distintivo_adición—, —
    —nulo—)
▶ —nombres_columnas_salida—, —sentencia_selección—, —archivo_mensajes—
    —nulo—)
▶▶

```

### Descripciones de parámetros

#### *nombre\_archivo*

Especifica la vía de acceso completa de un archivo de formas al que se van a exportar los datos especificados. Debe especificar un valor que no sea nulo para este parámetro.

Puede utilizar el procedimiento almacenado ST\_export\_shape para exportar un nuevo archivo o para exportar a un archivo existente añadiendo los datos exportados al mismo:

- Si está exportando a un nuevo archivo, puede especificar la extensión opcional de archivo como .shp o .SHP. Si especifica .shp o .SHP para la extensión del archivo, DB2 Spatial Extender crea el archivo con el valor *nombre\_archivo* especificado. Si no especifica la extensión opcional de archivo, DB2 Spatial Extender crea el archivo que tiene el nombre del valor *nombre\_archivo* que el usuario especifica con una extensión .shp.
- Si está exportando datos añadiendo datos a un archivo existente, DB2 Spatial Extender primero busca una coincidencia exacta del nombre que especifica para el parámetro *nombre\_archivo*. Si DB2 Spatial Extender no encuentra una coincidencia exacta, busca primero un archivo con extensión .shp y, a continuación, un archivo con la extensión .SHP.

Si el valor del parámetro *indicador\_adición* indica que no está realizando adiciones a un archivo existente, pero que el archivo nombrado en el parámetro *nombre\_archivo* ya existe, DB2 Spatial Extender devuelve un error y no sobrescribe el archivo.

Consulte Notas sobre el uso para ver una lista de los archivos que están grabados en la máquina del servidor. El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para crear los archivos y escribir en los mismos.

El tipo de datos de este parámetro es VARCHAR(256).

#### *distintivo\_adición*

Indica si los datos que se van a exportar se van a añadir a un archivo de formas existente. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Indique si desea realizar adiciones a un archivo de formas existente de la forma siguiente:

- Si desea añadir datos a un archivo de formas existente, especifique cualquier valor distinto de 0 (cero) y nulo. En este caso, la estructura del archivo debe coincidir con los datos exportados; en caso contrario, se devuelve un error.
- Si desea exportar a un nuevo archivo, especifique 0 (cero) o nulo. En este caso, DB2 Spatial Extender no sobrescribe ningún archivo existente.

El tipo de datos de este parámetro es SMALLINT.

#### *nombres\_columnas\_salida*

Especifica uno o varios nombres de columna (separados por comas) que se utilizarán para columnas no espaciales en el archivo dBASE de salida. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizan los nombres que se derivan de la sentencia SELECT.

Si especifica este parámetro pero no coloca los nombres de columna entre comillas dobles, los nombres de columna se convertirán a mayúsculas. El número de columnas especificadas debe coincidir con el número de columnas que se devuelven de la sentencia SELECT, según se especifica en el parámetro *sentencia\_selección*, excluyendo la columna espacial.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *sentencia\_selección*

Especifica el subelemento que devuelve los datos que se van a exportar. El subelemento debe hacer referencia exactamente a una columna espacial y a cualquier número de columnas de atributos. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *archivo\_mensajes*

Especifica el nombre de la vía de acceso completa del archivo (en el servidor) que contendrá mensajes sobre la operación de exportación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se creará ningún archivo para los mensajes de DB2 Spatial Extender.

Los mensajes que se envían a este archivo de mensajes pueden ser:

- Mensajes informativos, como, por ejemplo, un resumen de la operación de exportación
- Mensajes de error para datos que no se han podido exportar, por ejemplo, debido a sistemas distintos de coordenadas

El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para crear el archivo.

El tipo de datos de este parámetro es VARCHAR(256).

## **Parámetros de salida**

#### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

#### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## **Notas sobre el uso**

Puede exportar sólo una columna espacial cada vez.

El procedimiento almacenado ST\_export\_shape crea y graba los cuatro archivos siguientes:

- El archivo principal de formas (extensión .shp).
- El archivo de índices de formas (extensión .shx).
- Un archivo dBASE que contiene datos para columnas no espaciales (extensión .dbf). Este archivo se crea solamente si las columnas de atributos realmente necesitan ser exportadas
- Un archivo de proyección que especifica el sistema de coordenadas que está asociado a los datos espaciales, si el sistema de coordenadas no es igual a "SIN ESPECIFICAR" (extensión .prj). El sistema de coordenadas se obtiene del primer registro espacial. Se produce un error si los registros subsiguientes tienen sistemas de coordenadas diferentes.

La tabla siguiente describe cómo los tipos de datos de DB2 se almacenan en archivos de atributos dBASE. Los demás tipos de datos de DB2 no están soportados.

Tabla 27. Almacenamiento de tipos de datos de DB2 en archivos de atributos

Tipo SQL	Tipo .dbf	Tipo .dbf	Decimales .dbf	Comentarios
SMALLINT	N	6	0	
INTEGER	N	11	0	
BIGINT	N	20	0	
DECIMAL	N	precisión+2	scale	
REAL FLOAT(1) hasta FLOAT(24)	F	14	6	
DOUBLE FLOAT(25) hasta FLOAT(53)	F	19	9	
CHARACTER, VARCHAR, LONG VARCHAR y DATALINK	C	lon	0	longitud = 255
DATE	D	8	0	
TIME	C	8	0	
TIMESTAMP	C	26	0	

Se da soporte a todos los sinónimos para los tipos de datos y tipos diferenciados que están basados en los tipos listados en la tabla precedente.

### Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_export\_shape. Este ejemplo utiliza un mandato CALL de DB2 para exportar todas las filas de la tabla CUSTOMERS a un archivo de formas que se creará y se denominará /tmp/export\_file:

```
call db2gse.ST_export_shape('/tmp/export_file',0,NULL,
    'select * from customers','/tmp/export_msg',?,?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

## ST\_import\_shape

Utilice este procedimiento almacenado para importar un archivo de formas a una base de datos que está habilitada para operaciones espaciales.

El procedimiento almacenado puede funcionar de cualquiera de estas dos maneras, en función del parámetro *indicador\_creación\_tabla*:

- DB2 Spatial Extender puede crear una tabla que tiene una columna espacial y columnas de atributos, y a continuación puede cargar las columnas de la tabla con los datos de archivo.
- En caso contrario, los datos de formas y atributos se pueden cargar en una tabla existente que tenga una columna espacial y columnas de atributos que coincidan con los datos del archivo.

## Autorización

El propietario de la instancia de DB2 debe tener los privilegios necesarios en el servidor para leer los archivos de entrada y opcionalmente para escribir en archivos de errores. Los requisitos de autorizaciones adicionales varían en función de si está importando a una tabla existente o a una nueva tabla.

- **Cuando importe a una tabla existente**, el ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las siguientes autorizaciones o uno de los siguientes privilegios:
  - DATAACCESS
  - Privilegio CONTROL sobre la tabla o la vista
  - Privilegio INSERT y SELECT en la tabla o vista
- **Cuando importe a una nueva tabla**, el ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las siguientes autorizaciones o uno de los siguientes privilegios:

- DBADM
- Autorización CREATETAB sobre la base de datos

El ID de usuario debe tener también una de las autorizaciones siguientes:

- Autorización IMPLICIT\_SCHEMA sobre la base de datos, si el nombre de esquema de la tabla no existe
- Privilegio CREATEIN sobre el esquema, si el esquema de la tabla existe

## Sintaxis

```
▶▶db2gse.ST_import_shape(—nombre_archivo—, —columnas_atr_entrada—, —nulo—, —nombre_srs—, —esquema_tabla—, —nulo—, —nombre_tabla—, —columnas_atr_tabla—, —nulo—, —distintivo_tabla_creación—, —nulo—, —parámetros_creación_tabla—, —nulo—, —columna_espacial—, —esquema_tipo—, —nulo—, —nombre_tipo—, —nulo—, —longitud_en_línea—, —nulo—, —columna_id—, —nulo—, —columna_id_es_identidad—, —nulo—, —recuento_reinicios—, —nulo—, —ámbito_confirmación—, —nulo—, —archivo_excepciones—, —nulo—, —archivo_mensajes—)▶▶
```

## Descripciones de parámetros

### *nombre\_archivo*

Especifica el nombre de la vía de acceso completa del archivo de formas que se va a importar. Debe especificar un valor que no sea nulo para este parámetro.

Si especifica la extensión opcional de archivo, especifique .shp o .SHP. DB2 Spatial Extender primero busca una coincidencia exacta del nombre de archivo



especificado. Si DB2 Spatial Extender no encuentra una coincidencia exacta, busca primero un archivo con extensión .shp y, a continuación, un archivo con la extensión .SHP.

Consulte Notas sobre el uso para ver una lista de los archivos necesarios, que deben residir en la máquina del servidor. El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para leer los archivos.

El tipo de datos de este parámetro es VARCHAR(256).

#### *columnas\_atr\_entrada*

Especifica una lista de columnas de atributos para importar desde el archivo dBASE. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se importan todas las columnas. Si el archivo dBASE no existe, este parámetro debe ser una cadena de caracteres vacía o tener un valor nulo.

Para especificar un valor que no sea nulo para este parámetro, utilice una de las especificaciones siguientes:

- **Listar los nombres de columnas de atributos.** El ejemplo siguiente muestra cómo especificar una lista de los nombres de las columnas de atributos que se van a importar desde el archivo dBASE:

```
N(COLUMN1,COLUMN5,COLUMN3,COLUMN7)
```

Si un nombre de columna no está entre comillas dobles, se convierte a mayúsculas. Cada nombre de la lista debe estar separado por una coma. Los nombres resultantes deben coincidir exactamente con los nombres de columna en el archivo dBASE.

- **Listar los números de columnas de atributos.** El ejemplo siguiente muestra cómo especificar una lista de los números de las columnas de atributos que se van a importar desde el archivo dBASE:

```
P(1,5,3,7)
```

Las columnas están numeradas empezando por 1. Cada número de la lista debe estar separado por una coma.

- **Indicar que no se van a importar datos de atributos.** Especifique una cadena de caracteres vacía ("" ) para indicar explícitamente que DB2 Spatial Extender *no* debe importar datos de atributos.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *nombre\_srs*

Identifica el sistema de referencia espacial que se va a utilizar para las geometrías que se importan en la columna espacial. Debe especificar un valor que no sea nulo para este parámetro.

La columna espacial no estará registrada. El sistema de referencia espacial (SRS) debe existir antes de importar los datos. El proceso de importación no crea implícitamente el SRS, pero compara el sistema de coordenadas del SRS con el sistema de coordenadas que está especificado en el archivo .prj (si está disponible con el archivo de formas). El proceso de importación también verifica que las extensiones de los datos en el archivo de formas pueden estar representadas en dicho sistema de referencia espacial. Es decir, el proceso de importación verifica que las extensiones estén dentro de las coordenadas X, Y, Z y M mínimas y máximas posibles del SRS.

El valor de *nombre\_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *esquema\_tabla*

Especifica el esquema al que pertenece la tabla que está especificada en el parámetro *nombre\_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *nombre\_tabla*

Especifica el nombre no calificado de la tabla en la que se cargará el archivo de formas importado. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *columnas\_atr\_tabla*

Especifica los nombres de columna de la tabla donde se almacenarán los datos de atributos del archivo dBASE. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizan los nombres de las columnas en el archivo dBASE.

Si se especifica este parámetro, el número de nombres debe coincidir con el número de columnas que se importan desde el archivo dBASE. Si existe la tabla, las definiciones de columna deben coincidir con los datos entrantes. Consulte Notas sobre el uso para ver una explicación de cómo los tipos de datos de atributos se correlacionan con tipos de datos de DB2.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *distintivo\_tabla\_creación*

Especifica si el proceso de importación creará una nueva tabla. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo o cualquier otro valor distinto de 0 (cero), se creará una nueva tabla. (Si la tabla ya existe, se devuelve un error.) Si este parámetro es 0 (cero), no se creará ninguna tabla y ya tabla ya deberá existir.

El tipo de datos de este parámetro es INTEGER.

#### *parámetros\_creación\_tabla*

Especifica las opciones que se añadirán a la sentencia CREATE TABLE que crea una tabla en la que se importarán los datos. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se añadirá ninguna opción a la sentencia CREATE TABLE.

Para especificar las opciones CREATE TABLE, utilice la sintaxis de la sentencia CREATE TABLE de DB2. Por ejemplo, para especificar un espacio de tabla para crear tablas, especifique:

```
IN Nombre_ET INDEX IN Nombre_indice_ET LONG IN Nombre_ET_largo
```

El tipo de datos de este parámetro es VARCHAR(32K).

#### *columna\_espacial*

Nombre de la columna espacial de la tabla en la que se cargarán los datos de formas. Debe especificar un valor que no sea nulo para este parámetro.

Para una nueva tabla, este parámetro especifica el nombre de la nueva columna espacial que se creará. En caso contrario, este parámetro especifica el nombre de una columna espacial existente en la tabla.

El valor de *columna\_espacial* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *esquema\_tipo*

Especifica el nombre de esquema del tipo de datos espaciales (especificado por el parámetro *nombre\_tipo*) que se utilizará al crear una columna espacial en una nueva tabla. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de DB2GSE.

El valor de *esquema\_tipo* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *nombre\_tipo*

Especifica el tipo de datos que se utilizará para los valores espaciales. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el tipo de datos está determinado por el archivo de formas y pertenece a uno de los tipos siguientes:

- ST\_Point
- ST\_MultiPoint
- ST\_MultiLineString
- ST\_MultiPolygon

Tenga en cuenta que los archivos de formas, por definición, permiten la distinción sólo entre puntos y multipuntos, pero no entre polígonos y multipolígonos ni entre cadenas lineales y multilíneas.

Si está importando a una tabla que no existe aún, este tipo de datos también se utiliza para el tipo de datos de la columna espacial. En este caso, el tipo de datos también puede ser un supertipo de datos de ST\_Point, ST\_MultiPoint, ST\_MultiLineString o ST\_MultiPolygon.

El valor de *nombre\_tipo* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *longitud\_en\_línea*

Especifica, para una nueva tabla, el máximo número de bytes que se asignarán para la columna espacial dentro de la tabla. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se utilizará ninguna opción `INLINE LENGTH` explícita en la sentencia `CREATE TABLE` y no se utilizará implícitamente ningún valor por omisión de DB2.

Los registros espaciales que exceden este tamaño se almacenan separadamente en el espacio de tabla de LOB, al que puede ser más lento acceder.

Los tamaños típicos que son necesarios para varios tipos espaciales son los siguientes:

- **Un punto:** 292.
- **Multipunto, línea o polígono:** Un valor lo más alto posible. Tenga en cuenta el número total de bytes en una fila no puede exceder el límite para el tamaño de página del espacio de tabla para el que la tabla se ha creado.

Consulte la documentación de DB2 acerca de la sentencia CREATE TABLE de SQL para ver una descripción completa de este valor. Vea también el programa de utilidad db2dart para determinar el número de geometrías en línea para tablas existentes y la posibilidad de modificar la longitud lineal.

El tipo de datos de este parámetro es INTEGER.

#### *columna\_id*

Especifica la columna que se creará para contener un número único para cada fila de datos. (Las herramientas ESRI requieren una columna denominada SE\_ROW\_ID.) Los valores exclusivos para esta columna se generan automáticamente durante el proceso de importación. Aunque debe especificar un valor para este parámetro, el valor debe ser nulo si no existe ninguna columna (con un ID exclusivo en cada fila) en la tabla o si no está añadiendo una columna de este tipo a una tabla creada recientemente. Si este parámetro tiene un valor nulo, no se creará ni llenará ninguna columna con números exclusivos.

**Restricción:** No puede especificar un nombre *columna\_id* que coincida con el nombre de alguna columna en el archivo dBASE.

Los requisitos y efecto de este parámetro dependen de si la tabla ya existe.

- **Para una tabla existente**, el tipo de datos del parámetro *columna\_id* puede ser cualquier tipo de número entero (INTEGER, SMALLINT o BIGINT).
- **Para una nueva tabla que se creará**, la columna se añade a la tabla cuando el procedimiento almacenado la crea. La columna se definirá de la forma siguiente:

```
INTEGER NOT NULL PRIMARY KEY
```

Si el valor del parámetro *columna\_id\_es\_identidad* no es nulo ni es 0 (cero), la definición se expande de la forma siguiente:

```
INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY  
( START WITH 1 INCREMENT BY 1 )
```

EL valor de *columna\_id* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *columna\_id\_es\_identidad*

Indica si la *columna\_id* especificada se creará utilizando la cláusula IDENTITY. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro es 0 (cero) o nulo, la columna no se creará como la columna de identidad. Si el parámetro es cualquier valor distinto de 0 o de nulo, la columna se creará como la columna de identidad. Este parámetro se ignora para tablas que ya existen.

El tipo de datos de este parámetro es SMALLINT.

#### *recuento\_reinicios*

Especifica que se debe iniciar una operación de importación en el registro  $n + 1$ . Se saltan los  $n$  primeros registros. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se importarán todos los registros (empezando con el número de registro 1).

El tipo de datos de este parámetro es INTEGER.

#### *ámbito\_confirmación*

Especifica que se realizará un COMMIT después de que se importen  $n$  registros como mínimo. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero) y no se confirmará ningún registro.

El tipo de datos de este parámetro es INTEGER.

#### *archivo\_excepciones*

Especifica la vía de acceso completa de un archivo de formas en el que están almacenados los datos de formas que no se han podido importar. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si el parámetro tiene un valor nulo, no se creará ningún archivo.

Si especifica un valor para el parámetro e incluye la extensión opcional de archivo, especifique .shp o .SHP. Si la extensión del archivo tiene un valor nulo, se añadirá una extensión .shp.

El archivo de excepciones mantiene el bloque completo de filas para las que una única sentencia de insertar no ha sido satisfactoria. Por ejemplo, supongamos que una fila no se puede importar porque los datos de formas están codificados de forma incorrecta. Una única sentencia de insertar intenta importar 20 filas, incluyendo la que está en error. Debido al problema con la fila sencilla, todo el bloque de 20 filas se graba en el archivo de excepciones.

Los registros sólo se graban en el archivo de excepciones cuando estos registros se pueden identificar correctamente, como es el caso cuando el tipo de registro de forma no es válido. Algunos tipos de corrupciones en los datos de formas (archivos .shp) y el índice de formas (archivos .shx) no permiten identificar los registros adecuados. En este caso, no se graba ningún registro en el archivo de excepciones y se emite un mensaje de error para informar del problema.

Si especifica un valor para este parámetro, se crearán cuatro archivos en el servidor. Consulte Notas sobre el uso para ver una explicación de estos archivos. El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para crear los archivos. Si los archivos ya existen, el procedimiento almacenado devuelve un error.

El tipo de datos de este parámetro es VARCHAR(256).

#### *archivo\_mensajes*

Especifica el nombre de la vía de acceso completa del archivo (en el servidor) que contendrá mensajes sobre la operación de importación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si el parámetro tiene un valor nulo, no se creará ningún archivo para los mensajes de DB2 Spatial Extender.

Los mensajes que se graban en el archivo de mensajes pueden ser:

- Mensajes informativos, como, por ejemplo, un resumen de la operación de importación
- Mensajes de error para datos que no se han podido importar, por ejemplo, debido a sistemas distintos de coordenadas

Estos mensajes corresponden a los datos que se almacenan en el archivo de excepciones (identificado por el parámetro *archivo\_excepciones*).

El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para crear el archivo. Si el archivo ya existe, el procedimiento almacenado devolverá un error.

El tipo de datos de este parámetro es VARCHAR(256).

## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Notas sobre el uso

El procedimiento almacenado ST\_import\_shape utiliza de uno a cuatro archivos:

- El archivo principal de formas (extensión .shp). Este archivo es necesario.
- El archivo de índices de formas (extensión .shx). Este archivo es opcional. Si está presente, es posible que mejore el rendimiento de la operación de importación.
- Un archivo dBASE que contiene datos de atributos (extensión .dbf). Este archivo es necesario sólo si se van a importar datos de atributos.
- El archivo de proyección que especifica el sistema de coordenadas de los datos de formas (extensión .prj). Este archivo es opcional. Si este archivo está presente, el sistema de coordenadas que está definido en el mismo se compara con el sistema de coordenadas del sistema de referencia espacial que especifica el parámetro *id\_srs*.

La tabla siguiente describe cómo los tipos de datos de atributos dBASE se correlacionan con tipos de datos de DB2. Los demás tipos de datos de atributos no están soportados.

Tabla 28. Relación entre tipos de datos de DB2 y tipos de datos de atributos de dBASE

Tipo .dbf	longitud .dbf? (Ver nota)	decimales .dbf (Ver nota)	Tipo SQL	Comentarios
N	< 5	0	SMALLINT	
N	< 10	0	INTEGER	
N	< 20	0	BIGINT	
N	<i>lon</i>	<i>dec</i>	DECIMAL( <i>lon,dec</i> )	<i>lon</i> <32

Tabla 28. Relación entre tipos de datos de DB2 y tipos de datos de atributos de dBASE (continuación)

Tipo .dbf	longitud .dbf? (Ver nota)	decimales .dbf (Ver nota)	Tipo SQL	Comentarios
F	<i>lon</i>	<i>dec</i>	REAL	<i>lon</i> + <i>dec</i> < 7
F	<i>lon</i>	<i>dec</i>	DOUBLE	
C	<i>lon</i>		CHAR( <i>lon</i> )	
L			CHAR(1)	
D			DATE	

**Nota:** Esta tabla incluye las siguientes variables, las cuales están definidas ambas en la cabecera del archivo dBASE:

- *lon*, que representa la longitud total de la columna en el archivo dBASE. DB2 Spatial Extender utiliza este valor con dos propósitos:
  - Para definir la precisión para el tipo de datos DECIMAL de SQL o la longitud para el tipo de datos CHAR de SQL
  - Para determinar cuál de los tipos de número entero o de coma flotante se utilizará
- *dec*, que representa el número máximo de dígitos situados a la derecha de una coma decimal de la columna en el archivo dBASE. DB2 Spatial Extender utiliza este valor para definir la escala para el tipo de datos DECIMAL de SQL.

Por ejemplo, supongamos que el archivo dBASE contiene una columna de datos cuya longitud (*lon*) está definida como 20. Supongamos que el número de dígitos situados a la derecha de la coma decimal (*dec*) está definido como 5. Cuando DB2 Spatial Extender importa datos de esta columna, utiliza los valores de *len* y *dec* para obtener el siguiente tipo de datos de SQL: DECIMAL(20,5).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado `ST_import_shape`. Este ejemplo utiliza un mandato `CALL` de DB2 para importar un archivo de formas denominado `/tmp/officesShape` a la tabla denominada `OFFICES`:

```
call db2gse.ST_import_shape('/tmp/officesShape',NULL,'USA_SRS_1',NULL,
                            'OFFICES',NULL,0,NULL,'LOCATION',NULL,NULL,NULL,
                            NULL,NULL,NULL,NULL,'/tmp/import_msg',?,?)
```

Los dos signos de interrogación al final de este mandato `CALL` representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

## ST\_register\_geocoder

Utilice este procedimiento almacenado para registrar un geocodificador que no sea el geocodificador `DB2SE_USA_GEOCODER`, que se proporciona con DB2 Spatial Extender. El geocodificador `DB2SE_USA_GEOCODER` es registrado por DB2 Spatial Extender cuando se habilita la base de datos.

**Requisitos previos:** Antes de registrar un geocodificador:

- Asegúrese de que la función que ejecuta el geocodificador ya esté creada. Se puede registrar cada función de geocodificador como un geocodificador con un nombre de geocodificador identificado de forma exclusiva.
- Obtenga información del proveedor del geocodificador, por ejemplo:

- La sentencia de SQL que crea la función
- Los valores que se deben utilizar con los parámetros ST\_create\_srs de forma que se dé soporte a datos geométricos
- Información para registrar el geocodificador, por ejemplo:
  - Una descripción del geocodificador
  - Descripciones de los parámetros para el geocodificador
  - Los valores por omisión de los parámetros del geocodificador

El tipo de retorno de la función de geocodificador debe coincidir con el tipo de datos de la columna geocodificada. Los parámetros de geocodificación pueden ser un nombre de columna (denominada *columna de geocodificación*) que contiene datos que el geocodificador necesita. Por ejemplo, los parámetros del geocodificador pueden identificar direcciones o un valor de una significación especial para el geocodificador, tal como el grado mínimo de coincidencia. Si el parámetro de geocodificación es un nombre de columna, la columna debe estar en la misma tabla o vista que la columna geocodificada.

El tipo de retorno de la función de geocodificador sirve como el tipo de datos para la columna geocodificada. El tipo de retorno puede ser cualquier tipo de datos de DB2, tipo definido por el usuario o tipo estructurado. Si se devuelve un tipo definido por el usuario o un tipo estructurado, la función de geocodificador es responsable de devolver un valor válido del tipo de datos respectivo. Si la función de geocodificador devuelve valores de un tipo espacial, que es ST\_Geometry o uno de sus subtipos, la función de geocodificador es responsable de reestructurar una geometría válida. La geometría se debe representar utilizando un sistema de referencia espacial existente. La geometría es válida si al invocar a la función espacial ST\_IsValid para la geometría, el resultado es un 1. Los datos devueltos desde la función de geocodificador se actualizan o se insertan en la tabla geocodificada, en función de qué operación (INSERT o UPDATE) haya causado la generación del valor geocodificado.

Para saber si un geocodificador ya está registrado, consulte la vista de catálogo DB2GSE.ST\_GEOCODERS.

## Autorización

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización DBADM sobre la base de datos donde reside el geocodificador que registra este procedimiento almacenado.

## Sintaxis

```

▶▶ db2gse.ST_register_geocoder ( ( nombre_geocodificador ,
▶ esquema_función , nombre_función , nombre_específico ,
  nulo , nulo , nulo ,
▶ valores_parámetros_omisión , descripciones_parámetros ,
  nulo , nulo ,
▶ proveedor , descripción ) )

```



## Descripciones de parámetros

### *nombre\_geocodificador*

Identifica de forma exclusiva el geocodificador. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *esquema\_función*

Especifica el esquema para la función que implementa este geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, este valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la función.

El valor *esquema\_función* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_función*

Especifica el nombre no calificado de la función que implementa este geocodificador. La función ya debe estar creada y listada en SYSCAT.ROUTINES.

Para este parámetro, puede especificar un valor nulo si se ha especificado el parámetro *nombre\_específico*. Si no se ha especificado el parámetro *nombre\_específico*, el valor de *nombre\_función*, junto con el valor implícita o explícitamente definido de *esquema\_función*, debe identificar de forma exclusiva la función. Si no se ha especificado el parámetro *nombre\_función*, DB2 Spatial Extender recupera el valor de *nombre\_función* de la vista de catálogo SYSCAT.ROUTINES.

El valor de *nombre\_función* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_específico*

Identifica el nombre específico de la función que implementa el geocodificador. La función ya debe estar creada y listada en SYSCAT.ROUTINES.

Para este parámetro, puede especificar un valor nulo si el parámetro *nombre\_función* está especificado y la combinación de *esquema\_función* y *nombre\_función* identifica de forma exclusiva la función de geocodificador. Si el nombre de la función de geocodificador está sobrecargado, el parámetro *nombre\_específico* no puede ser un valor nulo. (Un nombre de función está *sobrecargado* si tiene el mismo nombre, pero no los mismos tipos de datos de parámetro o parámetros, que una o varias de las otras funciones.)

El valor de *nombre\_específico* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *valores\_parámetros\_omisión*

Especifica la lista de valores por omisión de parámetros de configuración para

la función de geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si todo el parámetro *valores\_parámetros\_predefinidos* tiene un valor nulo, todos los valores por omisión del parámetro tienen un valor nulo.

Si especifica algún valor de parámetro, especifíquelos en el orden en que la función los ha definido y sepárelos mediante una coma. Por ejemplo:

*valor\_parm1\_omisión,valor\_parm2\_omisión,...*

Cada valor de parámetro es una expresión de SQL. Siga estas directrices:

- Si un valor es una cadena de caracteres, póngalo entre comillas dobles.
- Si un valor de parámetro es un número, no lo ponga entre comillas simples.
- Si el valor del parámetro es un valor nulo, asígnelo al tipo correcto. Por ejemplo, en lugar de especificar simplemente NULL, especifique:  
CAST(NULL AS INTEGER)
- Si el parámetro de geocodificación va a ser una columna de geocodificación, no especifique el valor por omisión del parámetro.

Si algún valor de parámetro no está especificado (es decir, si especifica dos comas consecutivas (...,,...)), este parámetro se debe especificar al configurar la geocodificación o al ejecutar la geocodificación en modalidad de proceso por lotes con el parámetro *valores\_parámetros* de los procedimientos almacenados respectivos.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *descripciones\_parámetros*

Especifica la lista de descripciones de parámetros de configuración para la función de geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si todo el parámetro *descripciones\_parámetros* tiene un valor nulo, todas las descripciones del parámetro tendrán un valor nulo. Cada descripción de parámetro que especifica explica el significado y la utilización del parámetro, y puede tener una longitud máxima de 256. Las descripciones para los parámetros deben estar separadas por comas y deben aparecer en el orden de los parámetros como están definidos por la función. Si se utilizará una coma dentro de la descripción de un parámetro, ponga la cadena de caracteres entre comillas simples o dobles. Por ejemplo:

descripción,'descripción2, que contiene una coma',descripción3

El tipo de datos de este parámetro es VARCHAR(32K).

#### *proveedor*

Especifica el proveedor que ha implementado el geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se registra ninguna información sobre el proveedor que ha implementado el geocodificador.

El tipo de datos de este parámetro es VARCHAR(128).

#### *descripción*

Describe el geocodificador explicando su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro es nulo, no se graba ninguna información de descripción sobre el geocodificador.

**Recomendación:** Incluya la información siguiente:

- Nombre del sistema de coordenadas si se devolverán datos espaciales, como por ejemplo, texto convencional (WKT) o binario convencional (WKB)

- Sistema de referencia espacial, si se devolverán ST\_Geometry o alguno de sus subtipos
- Nombre del área geográfica a la que se aplica este geocodificador
- Cualquier otra información sobre el geocodificador que los usuarios deban saber

El tipo de datos de este parámetro es VARCHAR(256).

## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo presume que el usuario desea crear un geocodificador que toma la latitud y la longitud como datos de entrada y geocodifica en datos espaciales ST\_Point. Para hacer esto, cree primero una función denominada *lat\_long\_gc\_func*. A continuación, registre un geocodificador denominado *SAMPLEGC*, que utiliza la función *lat\_long\_gc\_func*.

Aquí tiene un ejemplo de la sentencia de SQL que crea la función *lat\_long\_gc\_func* que devuelve ST\_Point:

```
CREATE FUNCTION lat_long_gc_func(latitude double,
    longitude double, srId integer)
    RETURNS db2gse.ST_Point
    LANGUAGE SQL
    RETURN db2gse.ST_Point(latitude, longitude, srId)
```

Después de crear la función, puede registrarla como un codificador. Este ejemplo muestra cómo utilizar el mandato CALL del procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado *ST\_register\_geocoder* para registrar un geocodificador denominado *SAMPLEGC* con la función *lat\_long\_gc\_func*:

```
call db2gse.ST_register_geocoder ('SAMPLEGC',NULL,'LAT_LONG_GC_FUNC','',1'
    ,NULL,'My Company','Latitude/Longitude to
    ST_Point Geocoder'?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_register\_spatial\_column

Utilice este procedimiento almacenado para registrar una columna espacial y para asociar un sistema de referencia espacial (SRS) al mismo.

Cuando se procesa este procedimiento almacenado, la información sobre la columna espacial que se está registrando se añade a la vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS. Si se registra una columna espacial, se crea una restricción en la tabla, si es posible, para asegurar que todas las geometrías utilicen el SRS especificado.

### Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización DBADM sobre la base de datos que contiene la tabla a la que pertenece la columna espacial que se está registrando
- Privilegio CONTROL sobre esta tabla
- Privilegio ALTER sobre esta tabla

### Sintaxis

```
▶▶ db2gse.ST_register_spatial_column—(—esquema_tabla—, —nombre_tabla—▶▶  
                                          |  
                                          —nulo—)  
▶▶, —nombre_columna—, —nombre_srs—)▶▶
```

### Descripciones de parámetros

#### *esquema\_tabla*

Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre\_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *nombre\_tabla*

Especifica el nombre no calificado de la tabla o vista que contiene la columna que se está registrando. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *nombre\_columna*

Especifica la columna que se está registrando. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *nombre\_srs*

Especifica el sistema de referencia espacial que se utiliza para esta columna espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

## Parámetros de salida

#### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

#### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_register\_spatial\_column. Este ejemplo utiliza un mandato CALL de DB2 para registrar la columna espacial denominada LOCATION en la tabla denominada CUSTOMERS. Este mandato CALL especifica el valor del parámetro *nombre\_srs* como USA\_SRS\_1:

```
call db2gse.ST_register_spatial_column(NULL,'CUSTOMERS','LOCATION',  
    'USA_SRS_1',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_remove\_geocoding\_setup

Utilice este procedimiento almacenado para eliminar toda la información de configuración de geocodificación para la columna geocodificada.

Este procedimiento almacenado elimina de las vistas de catálogo DB2GSE.ST\_GEOCODING y DB2GSE.ST\_GEOCODING\_PARAMETERS información que está asociada con la columna geocodificada especificada.

**Restricción:**

No puede eliminar una configuración de geocodificación si la geocodificación automática está habilitada para la columna geocodificada.

**Autorización**

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización DATAACCESS sobre la base de datos que contiene la tabla sobre la que operará el geocodificador especificado
- Privilegio CONTROL sobre esta tabla
- Privilegio UPDATE sobre esta tabla

**Sintaxis**

```
►►db2gse.ST_remove_geocoding_setup(—esquema_tabla—, —nombre_tabla—, —  
                                          nulo                                          ) —  
►—nombre_columna—) —————►►
```

**Descripciones de parámetros**

*esquema\_tabla*

Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre\_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

*nombre\_tabla*

Especifica el nombre no calificado de la tabla o vista que contiene la columna en la que se van a insertar los datos geocodificados o en la que se van a actualizar los mismos. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

*nombre\_columna*

Especifica la columna en la que se van a insertar los datos geocodificados o se van a actualizar los mismos. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_remove\_geocoding\_setup. Este ejemplo utiliza un mandato CALL de DB2 para eliminar la configuración de geocodificación para la tabla denominada CUSTOMER y la columna denominada LOCATION:

```
call db2gse.ST_remove_geocoding_setup(NULL, 'CUSTOMERS', 'LOCATION',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_run\_geocoding

Utilice este procedimiento almacenado para ejecutar un geocodificador en modalidad de proceso por lotes en una columna geocodificada.

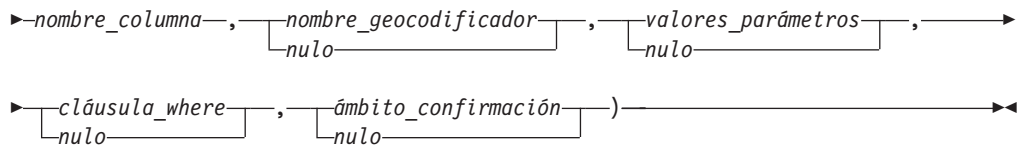
### Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización DATAACCESS sobre la base de datos que contiene la tabla sobre las que operará el geocodificador especificado
- Privilegio CONTROL sobre esta tabla
- Privilegio UPDATE sobre esta tabla

### Sintaxis

```
►► db2gse.ST_run_geocoding—(—esquema_tabla—,—nombre_tabla—,——————►  
                                  └─nulo—┘
```



## Descripciones de parámetros

### *esquema\_tabla*

Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre\_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_tabla*

Especifica el nombre no calificado de la tabla o vista que contiene la columna en la que se van a insertar los datos geocodificados o en la que se van a actualizar los mismos. Si hay especificado un nombre de vista, la vista debe ser una vista actualizable. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_columna*

Especifica la columna en la que se van a insertar los datos geocodificados o se van a actualizar los mismos. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_geocodificador*

Especifica el geocodificador que va a realizar la geocodificación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, la geocodificación la realiza el geocodificador que se ha especificado al configurar la geocodificación.

El valor de *nombre\_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *valores\_parámetros*

Especifica la lista de valores de parámetros de configuración para la función de geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si todo el parámetro *valores\_parámetros* tiene un valor nulo, los valores que se utilizan son los valores de los parámetros que se han



especificado al configurar el geocodificador o los valores de los parámetros por omisión para el geocodificador si el geocodificador no se ha configurado.

Si especifica algún valor de parámetro, especifíquelos en el orden en que la función los ha definido y sepárelos mediante una coma. Por ejemplo:

*parámetro1-valor,parámetro2-valor,...*

Cada valor de parámetro puede ser un nombre de columna, una cadena de caracteres, un valor numérico o un valor nulo.

Cada valor de parámetro es una expresión de SQL. Siga estas directrices:

- Si un valor de parámetro es un nombre de columna de geocodificación, asegúrese de que la columna esté en la misma tabla o vista donde está ubicada la columna codificada.
- Si un valor de parámetro es una cadena de caracteres, póngalo entre comillas simples.
- Si un valor de parámetro es un número, no lo ponga entre comillas simples.
- Si el parámetro tiene un valor nulo, asígnelo al tipo correcto. Por ejemplo, en lugar de especificar simplemente NULL, especifique:

CAST(NULL AS INTEGER)

Si algún valor de parámetro no está especificado (es decir, si especifica dos comas consecutivas (...,,...)), este parámetro se debe especificar al configurar la geocodificación o al ejecutar la geocodificación en modalidad de proceso por lotes con el parámetro *valores\_parámetros* de los procedimientos almacenados respectivos.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *cláusula\_where*

Especifica el cuerpo de la cláusula WHERE, donde define una restricción sobre el conjunto de registros que se van a geocodificar. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si el parámetro *cláusula\_where* tiene un valor nulo, el comportamiento resultante depende de si se ha configurado la geocodificación para la columna (especificada en el parámetro *nombre\_columna*) antes de ejecutar el procedimiento almacenado. Si el parámetro *cláusula\_where* tiene un valor nulo y:

- Se ha especificado un valor al configurar la geocodificación, dicho valor se utilizará para el parámetro *cláusula\_where*.
- No se ha configurado la geocodificación o no se ha especificado ningún valor al configurar la geocodificación, no se utilizará ninguna cláusula where.

Puede especificar una cláusula que se refiera a cualquier columna de la tabla o vista sobre la que vaya a operar el geocodificador. No especifique la palabra clave WHERE.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *ámbito\_confirmación*

Especifica que se va a realizar un COMMIT después de geocodificar *n* registros. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si el parámetro *ámbito\_confirmación* tiene un valor nulo, el comportamiento resultante depende de si se ha configurado la geocodificación para la columna

(especificada en el parámetro *nombre\_columna*) antes de ejecutar el procedimiento almacenado. Si el parámetro *ámbito\_confirmación* tiene un valor nulo y:

- Se ha especificado un valor al configurar la geocodificación para la columna, dicho valor se utilizará para el parámetro *ámbito\_confirmación*.
- No se ha configurado la geocodificación o se ha configurado pero no se ha especificado ningún valor, se utilizará el valor por omisión de 0 (cero) y no se realizará ningún COMMIT.

El tipo de datos de este parámetro es INTEGER.

## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_run\_geocoding. Este ejemplo utiliza un mandato CALL de DB2 para geocodificar la columna LOCATION en la tabla denominada CUSTOMER. Este mandato CALL especifica el valor del parámetro *nombre\_geocodificador* como DB2SE\_USA\_GEOCODER y el valor del parámetro *ámbito\_confirmación* como 10. Se va a realizar un COMMIT después de cada 10 registros geocodificados:

```
call db2gse.ST_run_geocoding(NULL, 'CUSTOMERS', 'LOCATION',  
    'DB2SE_USA_GEOCODER', NULL, NULL, 10, ?, ?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_setup\_geocoding

Utilice este procedimiento almacenado para asociar una columna que se va a geocodificar con un geocodificador y para configurar los parámetros de geocodificación correspondientes. La información que se configura aquí se registra en las vistas de catálogo DB2GSE.ST\_GEOCODING y DB2GSE.ST\_GEOCODING\_PARAMETERS.

Este procedimiento almacenado no invoca a la geocodificación. Proporciona una manera para que el usuario especifique los valores de los parámetros para la columna que se va a codificar. Con estos valores, se puede realizar con una interfaz mucho más sencilla la invocación subsiguiente de la geocodificación de proceso por lotes o la geocodificación automática. Los valores de los parámetros que se especifican en este paso de configuración alteran temporalmente cualquiera de los valores por omisión de los parámetros para el geocodificador que se han especificado cuando el geocodificador se ha registrado. También puede alterar temporalmente estos valores de parámetros ejecutando el procedimiento almacenado `ST_run_geocoding` en modalidad de proceso por lotes.

Este paso es un requisito necesario para la geocodificación automática. No puede habilitar la geocodificación automática sin configurar primero los parámetros de geocodificación. Este paso es un requisito necesario para la geocodificación de proceso por lotes. Puede ejecutar la geocodificación en modalidad de proceso por lotes con o sin realizar este paso de configuración. Sin embargo, si el paso de configuración se realiza antes de la geocodificación de proceso por lotes, los valores de los parámetros se toman del momento de configuración si no se han especificado en el momento de la ejecución.

## Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización `DATAACCESS` sobre la base de datos que contiene la tabla sobre la que operará el geocodificador especificado
- Privilegio `CONTROL` sobre esta tabla
- Privilegio `UPDATE` sobre esta tabla

## Sintaxis

```

▶▶ db2gse.ST_setup_geocoding ( ( esquema_tabla , nombre_tabla ,
                                └─┬─┘
                                nulo
▶ nombre_columna , nombre_geocodificador , valores_parámetros ,
                                └─┬─┘
                                nulo
▶ columnas_geocodificación_automática , cláusula_where ,
    └─┬─┘                               └─┬─┘
    nulo                               nulo
▶ ámbito_confirmación )
    └─┬─┘
    nulo

```

## Descripciones de parámetros

### *esquema\_tabla*

Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre\_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial `CURRENT SCHEMA` se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *nombre\_tabla*

Especifica el nombre no calificado de la tabla o vista que contiene la columna en la que se van a insertar los datos geocodificados o en la que se van a actualizar los mismos. Si hay especificado un nombre de vista, la vista debe ser actualizable. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *nombre\_columna*

Especifica la columna en la que se van a insertar los datos geocodificados o se van a actualizar los mismos. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *nombre\_geocodificador*

Especifica el geocodificador que va a realizar la geocodificación. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

#### *valores\_parámetros*

Especifica la lista de valores de parámetros de configuración para la función de geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si todo el parámetro *valores\_parámetros* tiene un valor nulo, los valores que se utilizan se toman de los valores por omisión que tenían los parámetros cuando se registró el geocodificador.

Si especifica valores de parámetros, especifíquelos en el orden en que la función los ha definido y sepárelos mediante una coma. Por ejemplo:

*parámetro1-valor,parámetro2-valor,...*

Cada valor de parámetro es una expresión de SQL y puede ser un nombre de columna, una cadena de caracteres, un valor numérico o un valor nulo. Siga estas directrices:

- Si un valor de parámetro es un nombre de columna de geocodificación, asegúrese de que la columna esté en la misma tabla o vista donde está ubicada la columna codificada.
- Si un valor de parámetro es una cadena de caracteres, póngalo entre comillas simples.
- Si un valor de parámetro es un número, no lo ponga entre comillas simples.
- Si el valor del parámetro está especificado como un valor nulo, asígnelo al tipo correcto. Por ejemplo, en lugar de especificar simplemente NULL, especifique:

CAST(NULL AS INTEGER)

Si algún valor de parámetro no está especificado (es decir, si especifica dos comas consecutivas (...)), este parámetro se debe especificar al configurar la geocodificación o al ejecutar la geocodificación en modalidad de proceso por lotes con el parámetro *valores\_parámetros* de los procedimientos almacenados respectivos.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *columnas\_geocodificación\_automática*

Especifica la lista de nombres de columna en la que se creará el desencadenante. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo y la geocodificación está habilitada, una actualización de cualquier columna de la tabla hace que se active el desencadenante.

Si especifica un valor para el parámetro *columnas\_geocodificación\_automática*, especifique nombres de columna en cualquier orden y separe los nombres de columna con una coma. El nombre de columna debe existir en la misma tabla donde se ubica la columna geocodificada.

El valor de este parámetro se aplica solamente a la geocodificación subsiguiente.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *cláusula\_where*

Especifica el cuerpo de la cláusula WHERE, donde define una restricción sobre el conjunto de registros que se van a geocodificar. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se define ninguna restricción en la cláusula WHERE.

La cláusula puede hacer referencia a cualquier columna de la tabla o vista sobre la que va a operar el geocodificador. No especifique la palabra clave WHERE.

El valor de este parámetro se aplica solamente a la geocodificación en modalidad de proceso por lotes subsiguiente.

El tipo de datos de este parámetro es VARCHAR(32K).

#### *ámbito\_confirmación*

Especifica que se va a realizar un COMMIT después de codificar *n* registros. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se realiza un COMMIT después de que se geocodifiquen todos los registros.

El valor de este parámetro se aplica solamente a la geocodificación en modalidad de proceso por lotes subsiguiente.

El tipo de datos de este parámetro es INTEGER.

## **Parámetros de salida**

#### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

*texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_setup\_geocoding. Este ejemplo utiliza un mandato CALL de DB2 para configurar un proceso de geocodificación para la columna geocodificada denominada LOCATION en la tabla denominada CUSTOMER. Este mandato CALL especifica el valor del parámetro *nombre\_geocodificador* como DB2SE\_USA\_GEOCODER:

```
call db2gse.ST_setup_geocoding(NULL, 'CUSTOMERS', 'LOCATION',
'DB2SE_USA_GEOCODER', 'ADDRESS,CITY,STATE,ZIP',1,100,80,,,'$HOME/sql1lib/
gse/refdata/ky.edg','$HOME/sql1lib/samples/extenders/spatial/EDGESample.loc',
'ADDRESS,CITY,STATE,ZIP',NULL,10,?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_unregister\_geocoder

Utilice este procedimiento almacenado para eliminar el registro de un geocodificador que no sea el geocodificador DB2SE\_USA\_GEOCODER, que se proporciona con DB2 Spatial Extender.

### Restricción:

No puede deshacer el registro de un geocodificador si está especificado en la configuración de geocodificación para cualquier columna.

Para determinar si un geocodificador está especificado en la configuración de geocodificación para una columna, compruebe las vista de catálogo DB2GSE.ST\_GEOCODING y DB2GSE.ST\_GEOCODING\_PARAMETERS. Para buscar información sobre el geocodificador del que desea deshacer el registro, consulte la vista de catálogo DB2GSE.ST\_GEOCODERS.

### Autorización

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización DBADM sobre la base de datos donde reside el geocodificador que se debe desregistrar.

### Sintaxis

```
►►—db2gse.ST_unregister_geocoder—(—nombre_geocodificador—)—————►►
```

## Descripciones de parámetros

### *nombre\_geocodificador*

Identifica de forma exclusiva el geocodificador. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado `ST_unregister_geocoder`. Este ejemplo utiliza un mandato `CALL` de DB2 para deshacer el registro del geocodificador denominado `SAMPLEGC`:

```
call db2gse.ST_unregister_geocoder('SAMPLEGC',?,?)
```

Los dos signos de interrogación al final de este mandato `CALL` representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

---

## ST\_unregister\_spatial\_column

Utilice este procedimiento almacenado para eliminar el registro de una columna espacial.

El procedimiento almacenado elimina el registro haciendo lo siguiente:

- Eliminando la asociación del sistema de referencia espacial con la columna espacial. La vista de catálogo `ST_GEOMETRY_COLUMNS` sigue conteniendo la columna espacial, pero la columna ya no está asociada con ningún sistema de referencia espacial.

- Para una tabla base, descartando la restricción que DB2 Spatial Extender ha puesto sobre esta tabla para asegurar que los valores de la geometría en esta columna espacial estén todos representados en el mismo sistema de referencia espacial.

## Autorización

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización DBADM
- Privilegio CONTROL sobre esta tabla
- Privilegio ALTER sobre esta tabla

## Sintaxis

```

▶▶ db2gse.ST_unregister_spatial_column—(—esquema_tabla—, —nombre_tabla—)
      |
      |—nulo—
      |
▶, —nombre_columna—)

```

## Descripciones de parámetros

### *esquema\_tabla*

Especifica el esquema al que pertenece la tabla que está especificada en el parámetro *nombre\_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_tabla*

Especifica el nombre no calificado de la tabla que contiene la columna que está especifica en el parámetro *nombre\_columna*. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

### *nombre\_columna*

Especifica la columna espacial de la que desea deshacer el registro. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre\_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).



## Parámetros de salida

### *código\_mje*

Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

### *texto\_mje*

Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

## Ejemplo

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST\_unregister\_spatial\_column. Este ejemplo utiliza un mandato CALL de DB2 para deshacer el registro de la columna espacial denominada LOCATION en la tabla denominada CUSTOMERS:

```
call db2gse.ST_unregister_spatial_column(NULL, 'CUSTOMERS', 'LOCATION', ?, ?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *código\_mje* y *texto\_mje*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.



---

## Capítulo 21. Vistas de catálogo

Vistas del catálogo de Spatial y Geodetic Data Management Feature suministran información de utilidad.

Las vistas de catálogo de Spatial Extender contienen información acerca de:

**“Vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS” en la página 251**  
Sistemas de coordenadas que puede utilizar

**“Vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS”**  
Columnas espaciales que puede llenar o actualizar

**“Vista de catálogo DB2GSE.ST\_GEOCODERS” en la página 254 y “Vista de catálogo DB2GSE.ST\_GEOCODING\_PARAMETERS” en la página 256**  
Geocodificadores que puede utilizar

**“Vista de catálogo DB2GSE.ST\_GEOCODING” en la página 255 y “Vista de catálogo DB2GSE.ST\_GEOCODING\_PARAMETERS” en la página 256**  
Especificaciones para configurar un geocodificador para que se ejecute automáticamente y para configurar, anticipadamente, operaciones que se deben realizar durante la geocodificación de proceso por lotes.

**“Vista de catálogo DB2GSE.ST\_SIZINGS” en la página 257**  
Longitudes máximas permitidas que puede asignar a las variables.

**“Vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS” en la página 258**  
Sistemas de referencia espacial que puede utilizar.

**“Vista de catálogo DB2GSE.ST\_UNITS\_OF\_MEASURE” en la página 261**  
Las unidades de medida (metros, millas, pies, etc.) en las que se pueden generar las funciones espaciales.

---

### Vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS

Utilice la vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS para buscar información sobre todas las columnas espaciales en todas las tablas que contienen datos espaciales en la base de datos.

Si se ha registrado una columna espacial en asociación con un sistema de referencia espacial, también puede utilizar la vista para saber el nombre y el identificador numérico del sistema de referencia espacial. Para obtener información adicional sobre las columnas espaciales, consulte la vista de catálogo SYSCAT.COLUMN de DB2.

Para ver una descripción de DB2GSE.ST\_GEOMETRY\_COLUMNS, consulte la tabla siguiente.

*Tabla 29. Columnas de la vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS*

Nombre	Tipo de datos	¿Anulable?	Contenido
TABLE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece la tabla que contiene esta columna espacial.
TABLE_NAME	VARCHAR(128)	No	Nombre no calificado de la tabla que contiene esta columna espacial.

Tabla 29. Columnas de la vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
COLUMN_NAME	VARCHAR(128)	No	Nombre de esta columna espacial.  La combinación de TABLE_SCHEMA, TABLE_NAME y COLUMN_NAME identifica exclusivamente la columna.
TYPE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece el tipo de datos declarado de esta columna espacial. Este nombre se obtiene del catálogo de DB2.
TYPE_NAME	VARCHAR(128)	No	Nombre no calificado del tipo de datos declarado de esta columna espacial. Este nombre se obtiene del catálogo de DB2.
SRS_NAME	VARCHAR(128)	Sí	Nombre del sistema de referencia espacial que está asociado con esta columna espacial. Si no hay ningún sistema de referencia espacial asociado a esta columna, SRS_NAME tiene un valor nulo.
SRS_ID	INTEGER	Sí	Identificador numérico del sistema de referencia espacial que está asociado a esta columna espacial. Si no hay ningún sistema de referencia espacial asociado a esta columna, SRS_ID tiene un valor nulo.

## Vista de catálogo DB2GSE.SPATIAL\_REF\_SYS

Cuando se crea el sistema de referencia espacial, DB2 Spatial Extender lo registra registrando su identificador y la información relacionada con el mismo en una tabla de catálogos. Las columnas seleccionadas de esta tabla comprenden la vista de catálogo DB2GSE.SPATIAL\_REF\_SYS, que se describe en la siguiente tabla.

Tabla 30. Columnas en la vista de catálogo DB2GSE.SPATIAL\_REF\_SYS

Nombre	Tipo de datos	¿Anulable?	Contenido
SRID	INTEGER	No	Identificador definido por el usuario para este sistema de referencia espacial.
SR_NAME	VARCHAR(64)	No	Nombre de este sistema de referencia espacial.
CSID	INTEGER	No	Identificador numérico para el sistema de coordenadas en el que se basa este sistema de referencia espacial.
CS_NAME	VARCHAR(64)	No	Nombre del sistema de coordenadas en el que se basa este sistema de referencia espacial.
AUTH_NAME	VARCHAR(256)	Sí	Nombre de la organización que establece los estándares para este sistema de referencia espacial.
AUTH_SRID	INTEGER	Sí	Identificador que la organización especificada en la columna AUTH_NAME asigna a este sistema de referencia espacial.
SRTEXT	VARCHAR(2048)	No	Texto de anotación para este sistema de referencia espacial.
FALSEX	FLOAT	No	Un número que, cuando se resta de un valor de coordenada X negativa, deja un número no negativo (es decir, un número positivo o cero).
FALSEY	FLOAT	No	Un número que, cuando se resta de un valor de coordenada Y negativa, deja un número no negativo (es decir, un número positivo o cero).

Tabla 30. Columnas en la vista de catálogo DB2GSE.SPATIAL\_REF\_SYS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
XYUNITS	FLOAT	No	Un número que, cuando se multiplique por una coordenada decimal X o una coordenada decimal Y, dé como resultado un número entero que se pueda guardar como un elemento de datos de 32 bits.
FALSEZ	FLOAT	No	Un número que, cuando se resta de un valor de coordenada Z negativa, deja un número no negativo (es decir, un número positivo o cero).
ZUNITS	FLOAT	No	Un número que, cuando se multiplique por una coordenada decimal Z, dé como resultado un número entero que se pueda guardar como un elemento de datos de 32 bits.
FALSEM	FLOAT	No	Un número que, cuando se resta de una medida negativa, deja un número no negativo (es decir, un número positivo o cero).
MUNITS	FLOAT	No	Un número que, cuando se multiplique por una medida decimal, dé como resultado un entero que se pueda guardar como un elemento de datos de 32 bits.

## Vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS

Consulte la vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS para recuperar información sobre sistemas de coordenadas registrados.

Spatial Extender registra automáticamente los sistemas de coordenadas en el catálogo de Spatial en las siguientes ocasiones:

- Cuando habilita una base de datos para operaciones espaciales.
- Cuando los usuarios definen sistemas de coordenadas adicionales en la base de datos.

Para ver una descripción de las columnas de esta vista, consulte la tabla siguiente.

Tabla 31. Columnas de la vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS

Nombre	Tipo de datos	¿Anulable?	Contenido
COORDSYS_NAME	VARCHAR(128)	No	Nombre de este sistema de coordenadas. El nombre es exclusivo dentro de la base de datos.
COORDSYS_TYPE	VARCHAR(128)	No	Tipo de este sistema de coordenadas: <b>PROYECTADO</b> Bidimensional. <b>GEOGRÁFICO</b> Tridimensional. Utiliza coordenadas X e Y. <b>GEOCÉNTRICO</b> Tridimensional. Utiliza coordenadas X, Y y Z. <b>SIN ESPECIFICAR</b> Sistema de coordenadas abstracto o que no es del mundo real.
DEFINITION	VARCHAR(2048)	No	El valor de esta columna se obtiene de la columna DEFINITION. Representación de WKT (texto convencional) de la definición de este sistema de coordenadas.

Tabla 31. Columnas de la vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
ORGANIZATION	VARCHAR(128)	Sí	Nombre de la organización (por ejemplo, un cuerpo de estándares como el ESPG (European Petrol Survey Group) que define este sistema de coordenadas.  Esta columna tiene un valor nulo si la columna ORGANIZATION_COORDSYS_ID tiene un valor nulo.
ORGANIZATION_COORDSYS_ID	INTEGER	Sí	Identificador numérico asignado a este sistema de coordenadas por la organización que ha definido el sistema de coordenadas. Este identificador y el valor de la columna ORGANIZATION identifican de forma exclusiva el sistema de coordenadas a menos que el identificador y el valor tengan ambos un valor nulo.  Si la columna ORGANIZATION tiene un valor nulo, la columna ORGANIZATION_COORDSYS_ID también tiene un valor nulo.
DESCRIPTION	VARCHAR(256)	Sí	Descripción del sistema de coordenadas que indica su aplicación.

## Vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS

Utilice la vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS para buscar información sobre todas las columnas espaciales en todas las tablas que contienen datos espaciales en la base de datos.

Si se ha registrado una columna espacial en asociación con un sistema de referencia espacial, también puede utilizar la vista para saber el nombre y el identificador numérico del sistema de referencia espacial. Para obtener información adicional sobre las columnas espaciales, consulte la vista de catálogo SYSCAT.COLUMN de DB2.

Para ver una descripción de DB2GSE.ST\_GEOMETRY\_COLUMNS, consulte la tabla siguiente.

Tabla 32. Columnas de la vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS

Nombre	Tipo de datos	¿Anulable?	Contenido
TABLE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece la tabla que contiene esta columna espacial.
TABLE_NAME	VARCHAR(128)	No	Nombre no calificado de la tabla que contiene esta columna espacial.
COLUMN_NAME	VARCHAR(128)	No	Nombre de esta columna espacial.  La combinación de TABLE_SCHEMA, TABLE_NAME y COLUMN_NAME identifica exclusivamente la columna.
TYPE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece el tipo de datos declarado de esta columna espacial. Este nombre se obtiene del catálogo de DB2.
TYPE_NAME	VARCHAR(128)	No	Nombre no calificado del tipo de datos declarado de esta columna espacial. Este nombre se obtiene del catálogo de DB2.

Tabla 32. Columnas de la vista de catálogo DB2GSE.ST\_GEOMETRY\_COLUMNS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
SRS_NAME	VARCHAR(128)	Sí	Nombre del sistema de referencia espacial que está asociado con esta columna espacial. Si no hay ningún sistema de referencia espacial asociado a esta columna, SRS_NAME tiene un valor nulo.
SRS_ID	INTEGER	Sí	Identificador numérico del sistema de referencia espacial que está asociado a esta columna espacial. Si no hay ningún sistema de referencia espacial asociado a esta columna, SRS_ID tiene un valor nulo.

## Vista de catálogo DB2GSE.ST\_GEOCODER\_PARAMETERS

Cuando habilita una base de datos para operaciones espaciales, la información sobre los parámetros del geocodificador proporcionado, DB2GSE\_USA\_GEOCODER, se registra automáticamente en el catálogo de DB2 Spatial Extender. Si registra geocodificadores adicionales, la información sobre los parámetros de los mismos también se registra en el catálogo. Para recuperar del catálogo la información sobre los parámetros de un geocodificador, consulte la vista de catálogo DB2GSE.ST\_GEOCODER\_PARAMETERS. Para ver una descripción de las columnas de esta vista, consulte la tabla siguiente.

Para obtener más información sobre los parámetros de los geocodificadores, consulte la vista de catálogo SYSCAT.ROUTINEPARMS de DB2. Para ver una descripción de esta vista, consulte el manual Consulta de SQL.

Tabla 33. Columnas de DB2GSE.ST\_GEOCODER\_PARAMETERS

Nombre	Tipo de datos	¿Anulable?	Contenido
GEOCODER_NAME	VARCHAR(128)	No	Nombre del geocodificador al que pertenece este parámetro.
ORDINAL	SMALLINT	No	Posición de este parámetro (es decir, el parámetro especificado en la columna PARAMETER_NAME) en la signatura de la función que sirve como el geocodificador especificado en la columna GEOCODER_NAME.  Los valores combinados en las columnas GEOCODER_NAME y ORDINAL identifican de forma exclusiva este parámetro.  Un registro en la vista de catálogo SYSCAT.ROUTINEPARMS de DB2 también contiene información sobre este parámetro. Este registro contiene un valor que aparece en la columna ORDINAL de SYSCAT.ROUTINEPARMS. Este valor es el mismo que el que aparece en la columna ORDINAL de la vista DB2GSE.ST_GEOCODER_PARAMETERS.
PARAMETER_NAME	VARCHAR(128)	Sí	Nombre de este parámetro. Si no se ha especificado un nombre cuando se ha creado la función a la que pertenece este parámetro, la columna PARAMETER_NAME tiene un valor nulo.  El contenido de la columna PARAMETER_NAME se obtiene del catálogo de DB2.

Tabla 33. Columnas de DB2GSE.ST\_GEOCODER\_PARAMETERS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
TYPE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece este parámetro. Este nombre se obtiene del catálogo de DB2.
TYPE_NAME	VARCHAR(128)	No	Nombre no calificado del tipo de datos de los valores asignados a este parámetro. Este nombre se obtiene del catálogo de DB2.
PARAMETER_DEFAULT	VARCHAR(2048)	Sí	El valor por omisión que se asignará a este parámetro. DB2 interpretará este valor como una expresión de SQL. Si el valor está entre comillas dobles, se enviará al geocodificador como una cadena de caracteres. En caso contrario, la evaluación de la expresión de SQL determinará qué tipo de datos de parámetros será cuando se envíe al geocodificador. Si la columna PARAMETER_DEFAULT contiene un valor nulo, este valor nulo se enviará al geocodificador.  El valor por omisión puede tener un valor correspondiente en la vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS. También puede tener un valor correspondiente en los datos de entrada del procedimiento almacenado ST_run_geocoding. Si cualquiera de los valores correspondientes difiere del valor por omisión, el valor correspondiente alterará temporalmente el valor por omisión.
DESCRIPTION	VARCHAR(256)	Sí	Descripción del parámetro que indica su aplicación.

## Vista de catálogo DB2GSE.ST\_GEOCODERS

Cuando habilita una base de datos para operaciones espaciales, el geocodificador proporcionado, DB2GSE\_USA\_GEOCODER, se registra automáticamente en el catálogo de DB2 Spatial Extender. Cuando desee hacer que geocodificadores adicionales estén disponibles para los usuarios, necesita registrar estos geocodificadores. Para recuperar información sobre los geocodificadores registrados, consulte la vista de catálogo DB2GSE.ST\_GEOCODERS. Para ver una descripción de las columnas de esta vista, consulte la tabla siguiente.

Para obtener información sobre los parámetros de los geocodificadores, consulte la vista de catálogo DB2GSE.ST\_GEOCODER\_PARAMETERS de DB2 Spatial Extender y la vista de catálogo SYSCAT.ROUTINEPARMS de DB2. Para obtener información sobre las funciones que se utilizan como geocodificadores, consulte la vista de catálogo SYSCAT.ROUTINES de DB2.

Tabla 34. Columnas de la vista de catálogo DB2GSE.ST\_GEOCODERS

Nombre	Tipo de datos	¿Anulable?	Contenido
GEOCODER_NAME	VARCHAR(128)	No	Nombre de este geocodificador. Es exclusivo en la base de datos.
FUNCTION_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece la función que está siendo utilizado como este geocodificador.
FUNCTION_NAME	VARCHAR(128)	No	Nombre no calificado de la función que está siendo utilizada como este geocodificador.



Tabla 34. Columnas de la vista de catálogo DB2GSE.ST\_GEOCODERS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
SPECIFIC_NAME	VARCHAR(128)	No	Nombre específico de la función que está siendo utilizado como este geocodificador.  Los valores combinados de FUNCTION_SCHEMA y SPECIFIC_NAME identifican de forma exclusiva la función que está siendo utilizada como este geocodificador.
RETURN_TYPE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece el tipo de datos del parámetro de salida de este geocodificador. Este nombre se obtiene del catálogo de DB2.
RETURN_TYPE_NAME	VARCHAR(128)	No	Nombre no calificado del tipo de datos del parámetro de salida de este geocodificador. Este nombre se obtiene del catálogo de DB2.
VENDOR	VARCHAR(256)	Sí	Nombre del proveedor que ha creado este geocodificador.
DESCRIPTION	VARCHAR(256)	Sí	Descripción del geocodificador que indica su aplicación.

## Vista de catálogo DB2GSE.ST\_GEOCODING

Cuando configura las operaciones de geocodificación, los detalles de la configuración se registran automáticamente en el catálogo de DB2 Spatial Extender. Para conocer estos detalles, consulte las vistas de catálogo DB2GSE.ST\_GEOCODING y DB2GSE.ST\_GEOCODING\_PARAMETERS. La vista de catálogo DB2GSE.ST\_GEOCODING, que se describe en la tabla siguiente, contiene detalles de todas las configuraciones; por ejemplo, el número de registros que un geocodificador procesará antes de cada confirmación. La vista de catálogo DB2GSE.ST\_GEOCODING\_PARAMETERS contiene detalles que son específicos para de geocodificador. Por ejemplo, las configuraciones para el geocodificador proporcionado, DB2GSE\_USA\_GEOCODER, incluyen el grado mínimo en el que deben coincidir las direcciones proporcionadas como datos de entrada y las direcciones reales para que el geocodificador geocodifique los datos de entrada. Este requisito mínimo, denominado grado mínimo de coincidencia se registra en la vista de catálogo DB2GSE.ST\_GEOCODING\_PARAMETERS.

Tabla 35. Columnas de la vista de catálogo DB2GSE.ST\_GEOCODING

Nombre	Tipo de datos	¿Anulable?	Contenido
TABLE_SCHEMA	VARCHAR(128)	No	Nombre del esquema que contiene la tabla que contiene la columna identificada en la columna COLUMN_NAME.
TABLE_NAME	VARCHAR(128)	No	Nombre no calificado de la tabla que contiene la columna identificada en la columna COLUMN_NAME.
COLUMN_NAME	VARCHAR(128)	No	Nombre de la columna espacial que se llenará según las especificaciones que se muestran en esta vista de catálogo.  Los valores combinados de las columnas TABLE_SCHEMA, TABLE_NAME y COLUMN_NAME identifican de forma exclusiva la columna espacial.

Tabla 35. Columnas de la vista de catálogo DB2GSE.ST\_GEOCODING (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
GEOCODER_NAME	VARCHAR(128)	No	Nombre del geocodificador que producirá datos para la columna espacial especificada en la columna COLUMN_NAME. Sólo se puede asignar un geocodificador a una columna espacial.
MODE	VARCHAR(128)	No	Modalidad para el proceso de geocodificación: <b>PROCESO POR LOTES</b> Sólo está habilitada la geocodificación de proceso por lotes. <b>AUTOMÁTICA</b> La geocodificación automática está configurada y activada. <b>NO VÁLIDA</b> Se ha detectado una inconsistencia en las tablas de catálogos espaciales; la entrada de geocodificación no es válida.
SOURCE_COLUMNS	VARCHAR(10000)	Sí	Nombres de las columnas de tabla configuradas para geocodificación automática. Siempre que se actualizan estas columnas, un desencadenante indica al geocodificador que geocodifique los datos actualizados.
WHERE_CLAUSE	VARCHAR(10000)	Sí	Condición de búsqueda dentro de una cláusula WHERE. Esta condición indica que cuando un geocodificador se ejecuta en modalidad de proceso por lotes, sólo geocodifica datos dentro del subconjunto especificado de registros.
COMMIT_COUNT	INTEGER	Sí	Número de filas que se procesarán durante la geocodificación de proceso por lotes antes de la emisión de una confirmación. Si el valor de la columna COMMIT_COUNT es 0 (cero) o nulo, no se emite ninguna confirmación.

## Vista de catálogo DB2GSE.ST\_GEOCODING\_PARAMETERS

Cuando se configuran las operaciones de geocodificación para un geocodificador determinado, los aspectos de la configuración específicos del geocodificador se registran automáticamente en el catálogo de Spatial Extender. Por ejemplo, una operación específica a un geocodificador determinado, DB2GSE\_USA\_GEOCODER, es comparar las direcciones proporcionadas como datos de entrada con unos datos de referencia y geocodificar las anteriores si coinciden con los últimos en un cierto grado, o en un grado superior al especificado. Cuando configura operaciones para este geocodificador, especifica cuál será este grado, denominado grado mínimo de coincidencia; y esta especificación se registra en el catálogo.

Para conocer los aspectos de configuración específicos del codificador para las operaciones de geocodificación, consulte la vista de catálogo DB2GSE.ST\_GEOCODING\_PARAMETERS. Esta vista se describe en la tabla siguiente.

Algunos valores por omisión para la definición de las operaciones de geocodificación están disponibles en la vista de catálogo DB2GSE.ST\_GEOCODER\_PARAMETERS. Los valores en la vista DB2GSE.ST\_GEOCODING\_PARAMETERS alteran temporalmente los valores por omisión.

Tabla 36. Columnas de la vista de catálogo DB2GSE.ST\_GEOCODING\_PARAMETERS

Nombre	Tipo de datos	¿Anulable?	Contenido
TABLE_SCHEMA	VARCHAR(128)	No	Nombre del esquema que contiene la tabla que contiene la columna identificada en la columna COLUMN_NAME.
TABLE_NAME	VARCHAR(128)	No	Nombre no calificado de la tabla que contiene la columna espacial.
COLUMN_NAME	VARCHAR(128)	No	Nombre de la columna espacial que se llenará según las especificaciones que se muestran en esta vista de catálogo.
ORDINAL	SMALLINT	No	<p>Los valores combinados en las columnas TABLE_SCHEMA, TABLE_NAME y COLUMN_NAME identifican de forma exclusiva esta columna espacial.</p> <p>Posición de este parámetro (es decir, el parámetro especificado en la columna PARAMETER_NAME) en la signatura de la función que sirve como geocodificador para la columna identificada en la columna COLUMN_NAME.</p> <p>Un registro en la vista de catálogo SYSCAT.ROUTINEPARMS de DB2 también contiene información sobre este parámetro. Este registro contiene un valor que aparece en la columna ORDINAL de SYSCAT.ROUTINEPARMS. Este valor es el mismo que el que aparece en la columna ORDINAL de la vista DB2GSE.ST_GEOCODING_PARAMETERS.</p>
PARAMETER_NAME	VARCHAR(128)	Sí	<p>Nombre de un parámetro en la definición del geocodificador. Si no se ha especificado ningún nombre al definir el geocodificador, PARAMETER_NAME tiene un valor nulo.</p> <p>Este contenido de la columna PARAMETER_NAME se obtiene del catálogo de DB2.</p>
PARAMETER_VALUE	VARCHAR(2048)	Sí	<p>El valor que se asigna a este parámetro. DB2 interpretará este valor como una expresión de SQL. Si el valor está entre comillas dobles, se enviará al geocodificador como una cadena de caracteres. En caso contrario, la evaluación de la expresión de SQL determinará qué tipo de datos de parámetros será cuando se envíe al geocodificador. Si la columna PARAMETER_VALUE contiene un valor nulo, este valor nulo se envía al geocodificador.</p> <p>La columna PARAMETER_VALUE corresponde a la columna PARAMETER_DEFAULT en la vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS. Si la columna PARAMETER_VALUE contiene un valor, este valor altera temporalmente el valor por omisión en la columna PARAMETER_DEFAULT. Si la columna PARAMETER_VALUE tiene un valor nulo, se utilizará el valor por omisión.</p>

## Vista de catálogo DB2GSE.ST\_SIZINGS

Utilice la vista de catálogo DB2GSE.ST\_SIZINGS para recuperar:

- Todas las variables soportadas por Spatial Extender; por ejemplo, nombre de sistema de coordenadas, nombre de geocodificador y variables a las que se puede asignar representaciones WKT de datos espaciales.
- La longitud máxima permitida, si se conoce, de valores asignados a estas variables (por ejemplo, las longitudes máximas permitidas de nombres de sistemas de coordenadas, de nombres de geocodificadores y de representaciones WKT de datos espaciales).

Para ver una descripción de las columnas en la vista, consulte la tabla siguiente.

Tabla 37. Columnas de la vista de catálogo DB2GSE.ST\_SIZINGS

Nombre	Tipo de datos	¿Anulable?	Contenido
VARIABLE_NAME	VARCHAR(128)	No	Término que indica una variable. El término es exclusivo dentro de la base de datos.
SUPPORTED_VALUE	INTEGER	Sí	Longitud máxima permitida de los valores asignados a la variable mostrada en la columna VARIABLE_NAME. Los valores posibles de la columna SUPPORTED_VALUE son:  <b>Un valor numérico distinto 0</b> La longitud máxima permitida de valores asignados a esta variable.  <b>0</b> Se permite cualquier longitud o no se puede determinar la longitud permitida.  <b>NULL</b> Spatial Extender no da soporte a esta variable.
DESCRIPTION	VARCHAR(128)	Sí	Descripción de esta variable.

## Vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS

Consulte la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS para recuperar información sobre los sistemas de referencia espacial registrados. Spatial Extender registra automáticamente los sistemas de referencia espacial en el catálogo de Spatial Extender en las siguientes ocasiones:

- Cuando habilita una base de datos para operaciones espaciales, cinco sistemas de referencia espacial geodésicos por omisión y 318 sistemas de referencia espacial geodésicos predefinidos.
- Cuando los usuarios crean sistemas de referencia espacial adicionales.

Para obtener el máximo partido de la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, necesita entender que cada sistema de referencia espacial está asociado a un sistema de coordenadas. El sistema de referencia espacial está diseñado en parte para convertir coordenadas derivadas del sistema de coordenadas en valores que DB2 pueda procesar con la máxima eficacia, y en parte para definir la extensión máxima posible de espacio a las que estas coordenadas pueden hacer referencia.

Para saber el nombre y el tipo de sistema de coordenadas asociado a un sistema de referencia espacial determinado, consulte las columnas COORDSYS\_NAME y COORDSYS\_TYPE de la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS. Para obtener más información acerca del sistema de coordenadas, consulte la vista de catálogo DB2GSE.ST\_COORDINATE\_SYSTEMS.

Tabla 38. Columnas de la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS

Nombre	Tipo de datos	¿Anulable?	Contenido
SRS_NAME	VARCHAR(128)	No	Nombre del sistema de referencia espacial. Este nombre es exclusivo dentro de la base de datos.
SRS_ID	INTEGER	No	Identificador numérico del sistema de referencia espacial. Cada sistema de referencia espacial tiene un identificador numérico exclusivo. Los sistemas de referencia espacial tienen valores SRS_ID comprendidos entre 2000000000 y 2000001000.
X_OFFSET	DOUBLE	No	Las funciones espaciales especifican sistemas de referencia por sus identificadores numéricos en lugar de hacerlo por los nombres. El desplazamiento se puede restar a todas las coordenadas X de una geometría. La sustracción es un paso en el proceso de convertir las coordenadas de la geometría en valores que DB2 pueda procesar con la máxima eficacia. Un paso subsiguiente es multiplicar el número resultante de la resta por el factor de escala que se muestra en la columna X_SCALE.
X_SCALE	DOUBLE	No	El factor de escala por el que se multiplica el número resultante de la operación de restar un desplazamiento a una coordenada X. Este factor es idéntico al valor que se muestra en la columna Y_SCALE.
Y_OFFSET	DOUBLE	No	El desplazamiento que se debe restar a todas las coordenadas Y de una geometría. La sustracción es un paso en el proceso de convertir las coordenadas de la geometría en valores que DB2 pueda procesar con la máxima eficacia. Un paso subsiguiente es multiplicar el número resultante de la sustracción por el factor de escala que se muestra en la columna Y_SCALE.
Y_SCALE	DOUBLE	No	Factor de escala por el que se multiplica el número resultante de la operación de restar un desplazamiento a una coordenada Y. Este factor es idéntico al valor que se muestra en la columna X_SCALE.
Z_OFFSET	DOUBLE	No	Desplazamiento que se debe restar a todas las coordenadas Z de una geometría. La sustracción es un paso en el proceso de convertir las coordenadas de la geometría en valores que DB2 pueda procesar con la máxima eficacia. Un paso subsiguiente es multiplicar el número resultante de la suscripción por el factor de escala que se muestra en la columna Z_SCALE.
Z_SCALE	DOUBLE	No	Factor de escala por el que se multiplica el número resultante de la operación de restar un desplazamiento a una coordenada Z.
M_OFFSET	DOUBLE	No	Desplazamiento que se debe restar a todas las medidas asociadas a una geometría. La sustracción es un paso en el proceso de convertir las medidas en valores que DB2 pueda procesar con la máxima eficacia. Un paso subsiguiente es multiplicar el número resultante de la sustracción por el factor de escala que se muestra en la columna M_SCALE.

Tabla 38. Columnas de la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
M_SCALE	DOUBLE	No	Factor de escala por el que se multiplica el número resultante de sustraer el desplazamiento a una medida.
MIN_X	DOUBLE	No	Mínimo valor posible para coordenadas X en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores de las columnas X_OFFSET y X_SCALE.
MAX_X	DOUBLE	No	Máximo valor posible para las coordenadas X en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores de las columnas X_OFFSET y X_SCALE.
MIN_Y	DOUBLE	No	Mínimo valor posible para las coordenadas Y en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas Y_OFFSET y Y_SCALE.
MAX_Y	DOUBLE	No	Máximo valor posible para las coordenadas Y en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas Y_OFFSET y Y_SCALE.
MIN_Z	DOUBLE	No	Mínimo valor posible para las coordenadas Z en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas Z_OFFSET y Z_SCALE.
MAX_Z	DOUBLE	No	Máximo valor posible para las coordenadas Z en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas Z_OFFSET y Z_SCALE.
MIN_M	DOUBLE	No	Mínimo valor posible para las medidas que se pueden almacenar con geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas M_OFFSET y M_SCALE.
MAX_M	DOUBLE	No	Máximo valor posible para las medidas que se pueden almacenar con las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas M_OFFSET y M_SCALE.
COORDSYS_NAME	VARCHAR(128)	No	Nombre identificador del sistema de coordenadas en las que se basa este sistema de referencia espacial.
COORDSYS_TYPE	VARCHAR(128)	No	Tipo de sistema de coordenadas en el que se basa este sistema de referencia espacial.
ORGANIZATION	VARCHAR(128)	Sí	Nombre de la organización (por ejemplo, un cuerpo de estándares) que ha definido el sistema de coordenadas en el que se basa este sistema de referencia espacial. ORGANIZATION es nulo si ORGANIZATION_COORSYS_ID es nulo.
ORGANIZATION_COORSYS_ID	INTEGER	Sí	Nombre de la organización (por ejemplo, un cuerpo de estándares) que ha definido el sistema de coordenadas en el que se basa este sistema de referencia espacial. ORGANIZATION_COORSYS_ID es nulo si ORGANIZATION es nulo.
DEFINITION	VARCHAR(2048)	No	Representación de WKT (texto convencional) de la definición de este sistema de coordenadas.
DESCRIPTION	VARCHAR(256)	Sí	Descripción del sistema de referencia espacial.

---

## Vista de catálogo DB2GSE.ST\_UNITS\_OF\_MEASURE

Algunas funciones espaciales aceptan o devuelven valores que indican una distancia determinada. En algunos casos, puede seleccionar en que unidad de medida se expresará la distancia. Por ejemplo, ST\_Distance devuelve la distancia mínima entre dos geometrías específicas. En alguna ocasión es posible que requiera que ST\_Distance devuelva la distancia en términos de millas; en otra ocasión, es posible que requiera una distancia expresada en términos de metros. Para saber las unidades de medida entre las que puede seleccionar, consulte la vista de catálogo DB2GSE.ST\_UNITS\_OF\_MEASURE.

Tabla 39. Columnas de la vista de catálogo DB2GSE.ST\_UNITS\_OF\_MEASURE

Nombre	Tipo de datos	¿Anulable?	Contenido
UNIT_NAME	VARCHAR(128)	No	Nombre de la unidad de medida. Este nombre es exclusivo en la base de datos.
UNIT_TYPE	VARCHAR(128)	No	Tipo de la unidad de medida. Los valores posibles son:  <b>LINEAL</b> La unidad de medida es lineal.  <b>ANGULAR</b> La unidad de medida es angular.
CONVERSION_FACTOR	DOUBLE	No	Valor numérico utilizado para convertir esta unidad de medida en su unidad base. La unidad base para las unidades lineales de medida es METER; la unidad base para unidades angulares de medida es RADIAN.  La unidad base propiamente dicha tiene un factor de conversión de 1.0.
DESCRIPTION	VARCHAR(256)	Sí	Descripción de la unidad de medida.

---





---

## Capítulo 22. Funciones espaciales: categorías y usos

Este capítulo proporciona información básica sobre todas las funciones espaciales y las organiza en categorías.

---

### Funciones espaciales: categorías y usos

DB2® Spatial Extender proporciona funciones que:

- Convierten geometrías entre varios formatos de intercambio de datos. Estas funciones se denominan funciones constructoras.
- Comparan geometrías para obtener información sobre perímetros, intersecciones y otros elementos. Estas funciones se denominan funciones de comparación.
- Devuelven información sobre las propiedades de las geometrías como, por ejemplo, las coordenadas y medidas de las geometrías, las relaciones entre geometrías y los perímetros.
- Generan nuevas geometrías a partir de geometrías existentes.
- Miden la distancia más corta entre puntos de geometrías.
- Proporcionan información sobre parámetros del índice.
- Proporcionan proyecciones y conversiones entre distintos sistemas de coordenadas.

---

### Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos

DB2 Spatial Extender proporciona funciones espaciales que convierten geometrías entre los formatos de intercambio de datos siguientes:

- Representación de texto convencional (well-known text, WKT)
- Representación binaria convencional (well-known binary, WKB)
- Representación de forma ESRI
- Representación GML (Geography Markup Language)

Las funciones para crear geometrías a partir de estos formatos se denominan *funciones de constructor*.

---

### Funciones de Constructor

DB2 Spatial Extender proporciona funciones espaciales que convierten geometrías entre los formatos de intercambio de datos siguientes:

- Representación de texto convencional (well-known text, WKT)
- Representación binaria convencional (well-known binary, WKB)
- Representación de forma ESRI
- Representación GML (Geography Markup Language)

Las funciones para crear geometrías a partir de estos formatos se denominan *funciones de constructor*. Las funciones constructoras tienen el mismo nombre que el tipo de datos de datos de geometría de la columna en la que se insertarán los

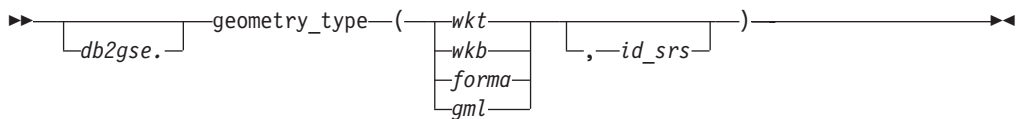
datos. Estas funciones operan de forma coherente sobre cada uno de los formatos de intercambio de datos proporcionados como entrada. Esta sección proporciona:

- El SQL para invocar a funciones que operan sobre formatos de intercambio de datos, y el tipo de geometría devuelto por estas funciones
- El SQL para invocar a una función que crea puntos a partir de coordenadas X e Y, y el tipo de geometría devuelto por esta función
- Ejemplos de código y conjuntos resultantes

## Funciones que operan sobre formatos de intercambio de datos

Esta sección proporciona la sintaxis para invocar a funciones que operan sobre formatos de intercambio de datos, describe los parámetros de entrada de las funciones e identifica el tipo de geometría devuelto por estas funciones.

### Sintaxis



### Parámetros y otros elementos de sintaxis

#### db2gse

Nombre del esquema al que pertenecen al que pertenecen los tipos de datos suministrados por DB2® Spatial Extender.

#### geometry\_type

Puede ser una de las funciones constructoras siguientes:

- ST\_Point
- ST\_LineString
- \
- ST\_Polygon
- ST\_MultiPoint
- ST\_MultiLineString
- ST\_MultiPolygon
- ST\_GeomCollection
- ST\_Geometry

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la geometría.

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la geometría.

**forma** Un valor de tipo BLOB(2G) que contiene la representación de forma ESRI de la geometría.

**gml** Valor de tipo CLOB(2G) que contiene la representación GML (Geography Markup Language) de la geometría.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

## Tipo de retorno

*geometry\_type*

Si *geometry\_type* es *ST\_Geometry*, el tipo dinámico devuelto corresponde a la geometría indicada por el valor de entrada.

Si *geometry\_type* es cualquier otro tipo, el tipo dinámico del tipo de geometría devuelto corresponde al nombre de la función. Si la geometría indicada por el valor de entrada no coincide con el nombre de función ni con el nombre de uno de sus subtipos, se emite un error.

## Una función que crea geometrías a partir de coordenadas

Esta sección proporciona la sintaxis para invocar a la función *ST\_Point*, una descripción de sus parámetros, e información sobre el tipo de geometría devuelto por la función.

La función *ST\_Point* crea geometrías no solo a partir de formatos de intercambio de datos, sino también a partir de valores de coordenadas: una función muy útil si los datos de ubicación ya están almacenados en la base de datos.

### Sintaxis

►►db2gse.ST\_Point(—| coordenadas | —, —*id\_srs*—)

#### coordenadas:

|—*coordenada\_x*—, —*coordenada\_y*— | —, —*coordenada\_z*— | —, —*coordenada\_m*— |

### Parámetros

#### *coordenada\_x*

Valor de tipo DOUBLE que especifica la coordenada X del punto resultante.

#### *coordenada\_y*

Valor de tipo DOUBLE que especifica la coordenada Y del punto resultante.

#### *coordenada\_z*

Valor de tipo DOUBLE que especifica la coordenada Z del punto resultante.

Si se omite el parámetro *coordenada\_z*, el punto resultante no tendrá coordenada Z. Para ese punto, el resultado de *ST\_Is3D* será un 0 (cero).

#### *coordenada\_m*

Valor de tipo DOUBLE que especifica la coordenada M del punto resultante.

Si se omite el parámetro *coordenada\_m*, el punto resultante no tendrá una medida. Para ese punto, el resultado de `ST_IsMeasured` será un 0 (cero).

**id\_srs** Valor de tipo `INTEGER` que identifica el sistema de referencia espacial correspondiente al punto resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo `DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS`, se emite una condición de excepción (`SQLSTATE 38SU1`).

## Tipo de retorno

`db2gse.ST_Point`

## Ejemplos

Esta sección proporciona ejemplos de código para invocar a funciones constructoras, código para crear tablas donde guardar los datos resultantes de las funciones constructoras, código para recuperar los datos resultantes y los propios datos resultantes.

El ejemplo siguiente inserta una fila en la tabla `SAMPLE_GEOMETRY` con un ID igual a 100 y un valor de punto con una coordenada X igual a 30, una coordenada Y igual a 40, dentro del sistema de referencia espacial 1 utilizando la representación de coordenadas y la representación de texto convencional (WKT). Luego inserta otra fila con un ID igual a 200 y un valor de cadena lineal con las coordenadas indicadas.

```
CREATE TABLE sample_geometry (id INT, geom db2gse.ST_Geometry);

INSERT INTO sample_geometry(id, geom)
VALUES(100,db2gse.ST_Geometry('point(30 40)', 1));

INSERT INTO sample_geometry(id, geom)
VALUES(200,db2gse.ST_Geometry('linestring(50 50, 100 100)', 1));

SELECT id, TYPE_NAME(geom) FROM sample_geometry

ID      2
-----
100    "ST_POINT"
200    "ST_LINESTRING"
```

Si sabemos que la columna espacial sólo puede contener valores de `ST_Point`, podemos utilizar el ejemplo siguiente para insertar dos puntos. Si se intenta insertar una cadena lineal o cualquier otro tipo distinto de un punto, se producirá un error de SQL. La primera inserción crea una geometría de punto a partir de la representación de texto convencional (WKT). La segunda inserción crea una geometría de punto a partir de valores de coordenadas numéricos. Observe que estos valores de entrada también se podrían seleccionar a partir de columnas de tabla existentes.

```
CREATE TABLE sample_points (id INT, geom db2gse.ST_Point);

INSERT INTO sample_points(id, geom)
VALUES(100,db2gse.ST_Point('point(30 40)', 1));

INSERT INTO sample_points(id, geom)
VALUES(101,db2gse.ST_Point(50, 50, 1));
```

```
SELECT id, TYPE_NAME(geom) FROM sample_geometry
```

```
ID      2
-----
100 "ST_POINT"
101 "ST_POINT"
```

El ejemplo siguiente utiliza SQL intercalado y supone que la aplicación llena las áreas de datos con los valores apropiados.

```
EXEC SQL BEGIN DECLARE SECTION;
    sqlint32 id = 0;
    SQL TYPE IS CLOB(10000) wkt_buffer;
    SQL TYPE IS CLOB(10000) gml_buffer;
    SQL TYPE IS BLOB(10000) wkb_buffer;
    SQL TYPE IS BLOB(10000) shape_buffer;
EXEC SQL END DECLARE SECTION;

// * Insertar aquí el código de aplicación para insertar datos */

EXEC SQL INSERT INTO sample_geometry(id, geom)
    VALUES(:id, db2gse.ST_Geometry(:wkt_buffer,1));

EXEC SQL INSERT INTO sample_geometry(id, geom)
    VALUES:id, db2gse.ST_Geometry(:wkb_buffer,1));

EXEC SQL INSERT INTO sample_geometry(id, geom)
    VALUES(:id, db2gse.ST_Geometry(:gml_buffer,1));

EXEC SQL INSERT INTO sample_geometry(id, geom)
    VALUES(:id, db2gse.ST_Geometry(:shape_buffer,1));
```

El siguiente código Java™ de ejemplo utiliza JDBC para insertar geometrías de punto utilizando los valores de coordenadas numéricas X, Y y utiliza la representación WKT para especificar las geometrías.

```
String ins1 = "INSERT into sample_geometry (id, geom)
    VALUES(?, db2gse.ST_PointFromText(CAST( ?
    as VARCHAR(128)), 1))";
PreparedStatement pstmt = con.prepareStatement(ins1);
pstmt.setInt(1, 100); // id value
pstmt.setString(2, "point(32.4 50.7)"); // wkt value
int rc = pstmt.executeUpdate();

String ins2 = "INSERT into sample_geometry (id, geom)
    VALUES(?, db2gse.ST_Point(CAST( ? as double),
    CAST(? as double), 1))";
pstmt = con.prepareStatement(ins2);
pstmt.setInt(1, 200); // id value
pstmt.setDouble(2, 40.3); // lat
pstmt.setDouble(3, -72.5); // long
rc = pstmt.executeUpdate();
```

---

## Conversión a una representación de texto convencional (WKT)

Las representaciones de texto son valores CLOB que representan cadenas de caracteres ASCII. Permiten el intercambio de geometrías en forma de texto ASCII.

La función **ST\_AsText** convierte en una cadena de caracteres WKT un valor de geometría contenido en una tabla. El ejemplo siguiente realiza una simple consulta desde la línea de mandatos para seleccionar valores que previamente se insertaron en la tabla SAMPLE\_GEOMETRY.

```
SELECT id, VARCHAR(db2gse.ST_AsText(geom), 50) AS WKTGEOM
FROM sample_geometry;
```

```
ID    WKTGEOM
-----
100   POINT ( 30.00000000 40.00000000)
200   LINESTRING ( 50.00000000 50.00000000, 100.00000000 100.00000000)
```

El ejemplo siguiente utiliza SQL intercalado para seleccionar los valores que previamente se insertaron en la tabla SAMPLE\_GEOMETRY.

```
EXEC SQL BEGIN DECLARE SECTION;
sqlint32 id = 0;
SQL TYPE IS CLOB(10000) wkt_buffer;
short wkt_buffer_ind = -1;
EXEC SQL END DECLARE SECTION;
```

```
EXEC SQL
SELECT id, db2gse.ST_AsText(geom)
INTO :id, :wkt_buffer :wkt_buffer_ind
FROM sample_geometry
WHERE id = 100;
```

Como alternativa, se puede utilizar el grupo de transformación ST\_WellKnownText para convertir implícitamente geometrías en su representación de texto convencional al realizar su vinculación. El código de ejemplo siguiente muestra cómo utilizar el grupo de transformación.

```
EXEC SQL BEGIN DECLARE SECTION;
sqlint32 id = 0;
SQL TYPE IS CLOB(10000) wkt_buffer;
short wkt_buffer_ind = -1;
EXEC SQL END DECLARE SECTION;
```

```
EXEC SQL
SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownText;
```

```
EXEC SQL
SELECT id, geom
INTO :id, :wkt_buffer :wkt_buffer_ind
FROM sample_geometry
WHERE id = 100;
```

La sentencia SELECT no utiliza ninguna función espacial para convertir la geometría.

Además de las funciones tratadas en esta sección, DB2® Spatial Extender proporciona otras funciones que también convierte geometrías hacia y desde representaciones de texto convencional. DB2 Spatial Extender proporciona estas otras funciones para implementar la especificación de OGC “Simple Features for SQL” y el estándar de ISO SQL/MM Part 3: Spatial. Estas funciones son:

- **ST\_WKTToSQL**
- **ST\_GeomFromText**
- **ST\_GeomCollFromTxt**
- **ST\_PointFromText**
- **ST\_LineFromText**
- **ST\_PolyFromText**
- **ST\_MPointFromText**
- **ST\_MLineFromText**
- **ST\_MPolyFromText**

---

## Conversión a una representación binaria convencional (WKB)

La representación WKB consta de estructuras de datos binarios que deben ser valores BLOB. Estos valores BLOB representan estructuras de datos binarios que deben ser gestionadas por un programa de aplicación escrito en un lenguaje de programación que DB2® soporte y para el que DB2 tenga una vinculación de lenguaje.

La función **ST\_AsBinary** convierte un valor de geometría contenido en una tabla en la representación binaria convencional (WKB) que se puede insertar en una variable BLOB de la memoria del programa. El ejemplo siguiente utiliza SQL intercalado para seleccionar los valores que previamente se insertaron en la tabla `SAMPLE_GEOMETRY`.

```
EXEC SQL BEGIN DECLARE SECTION;
    sqlint32 id = 0;
    SQL TYPE IS BLOB(10000) wkb_buffer;
    short wkb_buffer_ind = -1;
EXEC SQL END DECLARE SECTION;

EXEC SQL
    SELECT id, db2gse.ST_AsBinary(geom)
    INTO :id, :wkb_buffer :wkb_buffer_ind
    FROM sample_geometry
    WHERE id = 200;
```

Como alternativa, se puede utilizar el grupo de transformación `ST_WellKnownBinary` para convertir implícitamente geometrías en su representación binaria convencional al realizar su vinculación. El código de ejemplo siguiente muestra cómo utilizar este grupo de transformación.

```
EXEC SQL BEGIN DECLARE SECTION;
    sqlint32 id = 0;
    SQL TYPE IS BLOB(10000) wkb_buffer;
    short wkb_buffer_ind = -1;
EXEC SQL END DECLARE SECTION;

EXEC SQL
    SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary;

EXEC SQL
    SELECT id, geom
    INTO :id, :wkb_buffer :wkb_buffer_ind
    FROM sample_geometry
    WHERE id = 200;
```

La sentencia `SELECT` no utiliza ninguna función espacial para convertir la geometría.

Además de las funciones tratadas en esta sección, existen otras funciones que también convierten geometrías a partir de representaciones binarias convencionales. DB2 Spatial Extender proporciona estas otras funciones para implementar la especificación de OGC "Simple Features for SQL" y el estándar de ISO SQL/MM Part 3: Spatial. Estas funciones son:

- **ST\_WKBToSQL**
- **ST\_GeomFromWKB**
- **ST\_GeomCollFromWKB**
- **ST\_PointFromWKB**
- **ST\_LineFromWKB**

- **ST\_PolyFromWKB**
- **ST\_MPointFromWKB**
- **ST\_MLineFromWKB**
- **ST\_MPolyFromWKB**

---

## Conversión a la representación de forma ESRI

La representación de forma ESRI consiste de estructuras de datos binarios que deben ser gestionadas por un programa de aplicación escrito en un lenguaje soportado.

La función **ST\_AsShape** convierte un valor de geometría contenido en una tabla en la representación de forma ESRI que se puede insertar en una variable BLOB de la memoria del programa. El ejemplo siguiente utiliza SQL intercalado para seleccionar los valores que previamente se insertaron en la tabla `SAMPLE_GEOMETRY`.

```
EXEC SQL BEGIN DECLARE SECTION;
    sqlint32 id;
    SQL TYPE IS BLOB(10000) shape_buffer;
EXEC SQL END DECLARE SECTION;

EXEC SQL
    SELECT id, db2gse.ST_AsShape(geom)
    INTO :id, :shape_buffer
    FROM sample_geometry;
```

Como alternativa, se puede utilizar el grupo de transformación `ST_Shape` para convertir implícitamente geometrías en su representación de forma al realizar su vinculación. El código de ejemplo siguiente muestra cómo utilizar el grupo de transformación.

```
EXEC SQL BEGIN DECLARE SECTION;
    sqlint32 id = 0;
    SQL TYPE IS BLOB(10000) shape_buffer;
    short shape_buffer ind = -1;
EXEC SQL END DECLARE SECTION;

EXEC SQL
    SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

EXEC SQL
    SELECT id, geom
    FROM sample_geometry
    WHERE id = 300;
```

La sentencia `SELECT` no utiliza ninguna función espacial para convertir la geometría.

---

## Conversión a la representación GML (Geography Markup Language)

Las representaciones GML (Geography Markup Language) son cadenas de caracteres ASCII. Permiten el intercambio de geometrías en forma de texto ASCII.

La función **ST\_AsGML** convierte en una cadena de texto GML un valor de geometría contenido en una tabla. El ejemplo siguiente selecciona los valores que previamente se insertaron en la tabla `SAMPLE_GEOMETRY`. Los resultados que



aparecen en el ejemplo siguiente se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

```
SELECT id, VARCHAR(db2gse.ST_AsGML(geom), 500) AS GMLGEOM
FROM sample_geometry;
```

```
ID          GMLGEOM
-----
100 <gml:Point srsName="EPSG:4269">
    <gml:coord><gml:X>30</gml:X><gml:Y>40</gml:Y></gml:coord>
    </gml:Point>
200 <gml:LineString srsName="EPSG:4269">
    <gml:coord><gml:X>50</gml:X><gml:Y>50</gml:Y></gml:coord>
    <gml:coord><gml:X>100</gml:X><gml:Y>100</gml:Y></gml:coord>
    </gml:LineString>
```

Como alternativa, se puede utilizar el grupo de transformación ST\_GML para convertir implícitamente geometrías en su representación HTML al realizar su vinculación.

```
SET CURRENT DEFAULT TRANSFORM GROUP = ST_GML
```

```
SELECT id, geom AS GMLGEOM
FROM sample_geometry;
```

```
ID          GMLGEOM
-----
100 <gml:Point srsName="EPSG:4269">
    <gml:coord><gml:X>30</gml:X><gml:Y>40</gml:Y></gml:coord>
    </gml:Point>
200 <gml:LineString srsName="EPSG:4269">
    <gml:coord><gml:X>50</gml:X><gml:Y>50</gml:Y></gml:coord>
    <gml:coord><gml:X>100</gml:X><gml:Y>100</gml:Y></gml:coord>
    </gml:LineString>
```

La sentencia SELECT no utiliza ninguna función espacial para convertir la geometría.

---

## Funciones que comparan elementos geográficos

Determinadas funciones espaciales devuelven información sobre las maneras en que los elementos geográficos están relacionados entre sí o son equiparables el uno con el otro. Otras funciones espaciales devuelven información sobre si dos definiciones de sistemas de coordenadas o dos sistemas de referencia espacial son iguales. En todos los casos, la información devuelta es el resultado de comparar entre sí geometrías, definiciones de sistemas de coordenadas o sistemas de referencia espacial. Las funciones que proporcionan esta información se denominan funciones de comparación.

*Tabla 40. Funciones de comparación por el objetivo*

Objetivo	Funciones
Determinar si el interior de una geometría forma intersección con el interior de otra.	<ul style="list-style-type: none"> <li>• ST_Contains</li> <li>• ST_Within</li> </ul>
Devolver información sobre las intersecciones de geometrías.	<ul style="list-style-type: none"> <li>• ST_Crosses</li> <li>• ST_Intersects</li> <li>• ST_Overlaps</li> <li>• ST_Touches</li> </ul>

Tabla 40. Funciones de comparación por el objetivo (continuación)

Objetivo	Funciones
Determinar si el rectángulo más pequeño que engloba a una geometría forma intersección con el rectángulo más pequeño que engloba a otra geometría.	<ul style="list-style-type: none"> <li>• ST_EnvIntersects</li> <li>• ST_MBRIntersects</li> </ul>
Determinar si dos objetos son idénticos.	<ul style="list-style-type: none"> <li>• ST_Equals</li> <li>• ST_EqualCoordsys</li> <li>• ST_EqualSRS</li> </ul>
Determina si las geometrías comparadas cumplen las condiciones de la serie de la matriz patrón DE-9IM.	<ul style="list-style-type: none"> <li>• ST_Relate</li> </ul>
Comprueba si hay una intersección entre dos geometrías.	<ul style="list-style-type: none"> <li>• ST_Disjoint</li> </ul>

## Funciones de comparación

Las funciones de comparación de DB2<sup>®</sup> Spatial Extender devuelven un valor de 1 (uno) si una comparación cumple determinados criterios, un valor de 0 (cero) si una comparación no consigue cumplir los criterios y un valor de nulo si no se ha podido realizar la comparación. No se pueden realizar comparaciones si la operación de comparación no se ha definido para los parámetros de entrada o si cualquiera de los parámetros es nulo. Las comparaciones sí pueden realizarse si a los parámetros se asignan geometrías con tipos de datos o dimensiones diferentes.

El modelo Dimensionally Extended 9 Intersection Model (DE-9IM) es un método matemático que define la relación espacial entre geometrías de tipos y dimensiones diferentes. Este modelo expresa relaciones espaciales entre todos los tipos de geometrías en forma de intersecciones por pares de interiores, perímetros y exteriores, teniendo en cuenta la dimensión de las intersecciones resultantes.

Dadas las geometrías  $a$  y  $b$ :  $I(a)$ ,  $B(a)$  y  $E(a)$  representan el interior, perímetro y exterior de  $a$ , respectivamente.  $I(b)$ ,  $B(b)$ , y  $E(b)$  representan el interior, perímetro y exterior de  $b$ . Las intersecciones de  $I(a)$ ,  $B(a)$  y  $E(a)$  con  $I(b)$ ,  $B(b)$ , y  $E(b)$  producen una matriz de 3 por 3. Cada intersección puede dar lugar a geometrías de dimensiones diferentes. Por ejemplo, si la intersección de los perímetros de dos polígonos consta de un punto y una cadena lineal, la función `dim` devuelve la dimensión máxima: 1.

El resultado de la función `dim` es un valor de  $-1$ ,  $0$ ,  $1$  ó  $2$ . El  $-1$  corresponde al conjunto nulo o `dim(null)`, que se devuelve si no se ha encontrado ninguna intersección.

Los resultados devueltos por las funciones de comparación se pueden interpretar o verificar comparándolos con una matriz patrón que representa los valores aceptables para el modelo DE-9IM.

La matriz patrón contiene los valores aceptables para cada una de las celdas de la matriz de intersección. Los valores patrón posibles son:

- V** Debe existir una intersección;  $\text{dim} = 0, 1$  ó  $2$ .
- F** No debe existir una intersección;  $\text{dim} = -1$ .
- \*** No importa si existe o no una intersección;  $\text{dim} = -1, 0, 1$  ó  $2$ .

- 0 Debe existir una intersección y su dimensión exacta debe ser 0;  $\dim = 0$ .
- 1 Debe existir una intersección y su dimensión máxima debe ser 1;  $\dim = 1$ .
- 2 Debe existir una intersección y su dimensión máxima debe ser 2;  $\dim = 2$ .

Por ejemplo, la matriz patrón siguiente correspondiente a la función ST\_Within incluye los valores T, F y \*.

Tabla 41. Matriz de ST\_Within. Matriz patrón de la función ST\_Within para combinaciones geométricas.

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Interior Geometría a	V	*	F
Perím. Geometría a	*	*	F
Exterior Geometría a	*	*	*

La función ST\_Within devuelve un valor de 1 cuando los interiores de ambas geometrías forman intersección y cuando el interior o perímetro de *a* no forma intersección con el exterior de *b*. El resto de condiciones carece de importancia.

Cada función tiene como mínimo una matriz patrón, pero algunas necesitan más de una matriz para describir las relaciones de las diversas combinaciones de tipos de geometrías.

El modelo DE-9IM fue desarrollado por Clementini y Felice, que ampliaron dimensionalmente el Modelo de Intersección 9 de Egenhofer y Herring. El modelo DE-9IM es el resultado de la colaboración de cuatro autores (Clementini, Eliseo, Di Felice y van Osstroom), que publicaron el modelo en el trabajo "A Small Set of Formal Topological Relationships Suitable for End-User Interaction", D. Abel y B.C. Ooi (Ed.), *Advances in Spatial Database—Third International Symposium. SSD '93*. LNCS 692. Pp. 277-295. El modelo de intersección 9 creado por M. J. Egenhofer y J. Herring (Springer-Verlag Singapore [1993]) se publicó en "Categorizing binary topological relationships between regions, lines, and points in geographic databases," *Tech. Report, Department of Surveying Engineering* de la Universidad de Maine, Orono, ME 1991.

## Funciones de comparación espaciales

Las funciones de comparación son:

- ST\_Contains
- ST\_Crosses
- ST\_Disjoint
- ST\_EnvIntersects
- ST\_EqualCoordsys
- ST\_Equals
- ST\_EqualSRS
- ST\_Intersects
- ST\_MBRIntersects
- ST\_Overlaps
- ST\_Relate
- ST\_Touches

- ST\_Within

---

## Funciones que comprueban si una geometría contiene otra

ST\_Contains y ST\_Within ambas toman dos geometrías como parámetros de entrada y determinan si el interior de una geometría forma intersección con el interior de la otra. En términos más sencillos, ST\_Contains determina si la primera geometría proporcionada a la función engloba a la segunda geometría (es decir, si la primera contiene a la segunda). ST\_Within determina si la primera geometría está completamente dentro de la segunda.

### ST\_Contains

Utilice ST\_Contains para determinar si una geometría está completamente dentro de otra.

ST\_Contains devuelve un valor de 1 (uno) si la segunda geometría está completamente dentro de la primera. La función ST\_Contains devuelve el resultado opuesto exacto de la función ST\_Within.

La Figura 44 en la página 275 muestra ejemplos de ST\_Contains:

- Una geometría de multipuntos contiene un punto o geometrías de multipuntos cuando todos los puntos se encuentran dentro de la primera geometría.
- Una geometría de polígono contiene una geometría de multipunto cuando todos los puntos se encuentran en los límites del polígono o en el interior del polígono.
- Una geometría de cadena lineal contiene geometrías de cadena lineal, multipunto o punto cuando todos los puntos se encuentran dentro de la primera geometría.
- Una geometría de polígono contiene geometrías de polígono, cadena lineal o punto cuando la segunda geometría se encuentran en el interior del polígono.

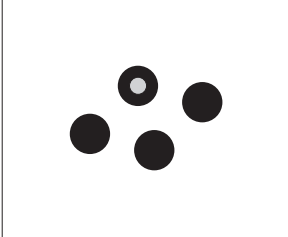
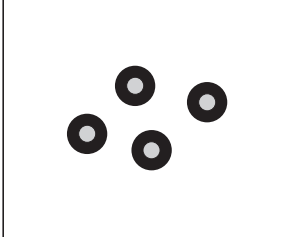
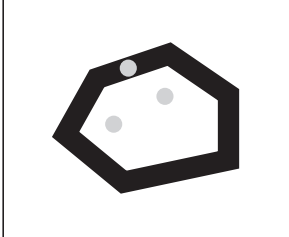
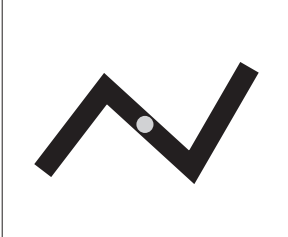
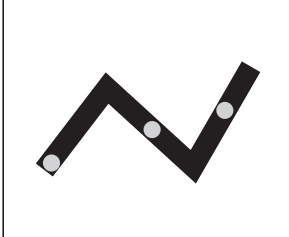
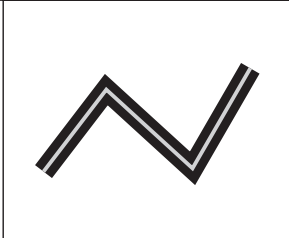
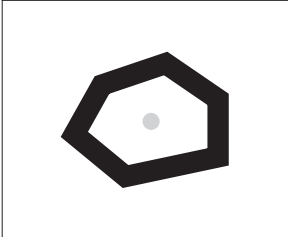
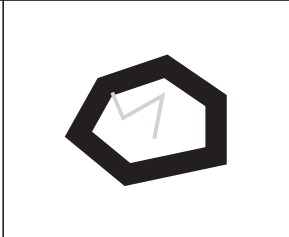
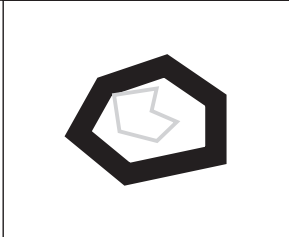
		
multipunto / punto	multipunto / multipunto	polígono / multipunto
		
cadena lineal / punto	cadena lineal / multipunto	cadena linea / cadena lineal
		
polígono / punto	polígono / cadena lineal	polígono / polígono

Figura 44. *ST\_Contains*. Las geometrías en negro representan la geometría *a* y las geometrías en gris representan la geometría *b*. En todos los casos, la geometría *a* contiene completamente a la geometría *b*.

La matriz patrón de la función *ST\_Contains* indica que los interiores de ambas geometrías deben formar intersección y que el interior o el perímetro de la secundaria (geometría *b*) no deben formar intersección con el exterior de la primaria (geometría *a*). El asterisco (\*) indica que no es significativo el hecho de que exista una intersección entre partes de las geometrías.

Tabla 42. Matriz de *ST\_Contains*

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Interior Geometría a	V	*	*
Perím. Geometría a	*	*	*
Exterior Geometría a	F	F	*

## ST\_Within

Utilice *ST\_Within* para determinar si una geometría está completamente dentro de otra.

*ST\_Within* devuelve un valor de 1 (uno) si la primera geometría está completamente dentro de la segunda. *ST\_Within* devuelve el resultado opuesto exacto de *ST\_Contains*.







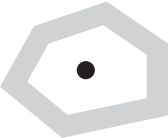


		
punto / multipunto	multipunto / multipunto	multipunto / polígono
		
punto / cadena lineal	multipunto / cadena lineal	cadena lineal / cadena lineal
		
punto / polígono	cadena lineal / polígono	polígono / polígono

Figura 45. ST\_Within

El matriz de patrón de la función ST\_Within indica que los interiores de ambas geometrías deben formar intersección y que el interior o el perímetro de la primera geometría (la geometría *a*) no debe formar intersección con el exterior de la secundaria *b*). El asterisco (\*) indica que las demás intersecciones no son significativas.

Tabla 43. Matriz de ST\_Within

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
<b>Interior Geometría a</b>	V	*	F
<b>Perím. Geometría a</b>	*	*	F
<b>Exterior Geometría a</b>	*	*	*

La Figura 45 muestra ejemplos de ST\_Within:

- Una geometría de punto está dentro de una geometría de multipunto cuando su interior forma intersección con uno de los puntos de la segunda geometría.
- Una geometría de multipunto está dentro de una geometría de multipunto cuando los interiores de todos los puntos forman intersección con la segunda geometría.
- Una geometría de multipunto está dentro de una geometría de polígono cuando todos los puntos se encuentran en los límites del polígono o en el interior del polígono.

- Una geometría de punto está dentro de una geometría de cadena lineal cuando todos los puntos están dentro de la segunda geometría. En la Figura 45 en la página 276, el punto no está dentro de la cadena lineal porque su interior no forma intersección con la cadena lineal; sin embargo, la geometría de multipunto está dentro de la cadena lineal porque todos sus puntos forman intersección con el interior de la cadena lineal.
- Una geometría de cadena lineal está dentro de otras geometrías de cadena lineal cuando todos sus puntos forman intersección con la segunda geometría.
- Una geometría de punto no está dentro de una geometría de polígono porque su interior no forma intersección con el perímetro o el interior del polígono.
- Una geometría de cadena lineal está dentro de una geometría de polígono cuando todos sus puntos forman intersección con el perímetro o el interior del polígono.
- Una geometría de polígono está dentro de una geometría de polígono cuando todos sus puntos forman intersección con el perímetro o el interior del polígono.

---

## Funciones que comprueban intersecciones entre geometrías

ST\_Intersects, ST\_Crosses, ST\_Overlaps y ST\_Touches determinan todos si una geometría forma intersección con otra. Difieren principalmente respecto al ámbito de intersección que es objeto de comprobación:

- ST\_Intersects prueba para determinar si las dos geometrías proporcionadas cumplen una de estas cuatro condiciones: los interiores de las geometrías forman intersección, sus perímetros forman intersección, el perímetro de la primera geometría forma intersección con el interior de la segunda o bien el interior de la primera geometría forma intersección con el perímetro de la segunda.
- ST\_Crosses se utiliza para analizar la intersección de geometrías de diferentes dimensiones, con una excepción: también puede analizar la intersección de cadenas lineales. En todos los casos, el propio lugar de intersección se considera que es una geometría; ST\_Crosses necesita que la dimensión de esta geometría sea menor que la de las geometrías que forman intersección (o bien, si ambas son cadenas lineales, que la dimensión del lugar de intersección sea menor que una cadena lineal). Por ejemplo, las dimensiones de una cadena lineal y un polígono son 1 y 2, respectivamente. Si estas geometrías forman intersección y el lugar de intersección es lineal (el recorrido de la cadena lineal a lo largo del polígono), entonces ese lugar se puede considerar que es una cadena lineal. Además, debido a que la dimensión de una cadena lineal (1) es menor que la de un polígono (2), ST\_Crosses, después de analizar la intersección, devolvería un valor de 1.
- Las geometrías proporcionadas a ST\_Overlaps como entrada deben tener la misma dimensión. ST\_Overlaps necesita que estas geometrías se solapen parcialmente, formando una nueva geometría (la región de solapamiento) que tiene la misma dimensión que las geometrías originales.
- ST\_Touches determina si los perímetros de las dos geometrías forman intersección.

### ST\_Intersects

Utilice ST\_Intersects para determinar si dos geometrías forman intersección.

ST\_Intersects devuelve un valor de 1 (uno) si la intersección no genera un conjunto vacío. ST\_Intersects devuelve el resultado opuesto exacto de ST\_Disjoint.

La función ST\_Intersects devuelve 1 (uno) si las condiciones de alguna de las siguientes matrices patrón generan un resultado TRUE (verdadero).

Tabla 44. Matriz de ST\_Intersects (1). La función ST\_Intersects devuelve 1 (uno) si los interiores de ambas geometrías forman intersección.

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	V	*	*
Exterior Geometría a	*	*	*

Tabla 45. Matriz de ST\_Intersects (2). La función ST\_Intersects devuelve 1 (uno) si el perímetro de la primera geometría forma intersección con el perímetro de la segunda geometría.

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	*	V	*
Exterior Geometría a	*	*	*

Tabla 46. Matriz de ST\_Intersects (3). La función ST\_Intersects devuelve 1 (uno) si el perímetro de la primera geometría forma intersección con el interior de la segunda.

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	V	*	*
Interior Geometría a	*	*	*
Exterior Geometría a	*	*	*

Tabla 47. Matriz de ST\_Intersects (4). La función ST\_Intersects devuelve 1 (uno) si los perímetros de cualquiera de las geometrías forman intersección.

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	V	*
Interior Geometría a	*	*	*
Exterior Geometría a	*	*	*

## ST\_Crosses

Utilice ST\_Crosses para determinar si una geometría se cruza con otra.

ST\_Crosses toma dos geometrías y devuelve un valor de 1 (uno) si:

- La intersección da lugar a una geometría cuya dimensión es menor que la dimensión máxima de las geometrías originales.
- El conjunto de intersección es interior respecto a ambas geometrías originales.

ST\_Crosses devuelve un valor nulo si la primera geometría es una superficie o multisuperficie o si la segunda geometría es un punto o multipunto. Para todas las demás combinaciones, ST\_Crosses devuelve un valor de 1 (que indica que las dos geometrías se cruzan) o un valor de 0 (que indica que no se cruzan).

La siguiente figura muestra multipuntos que se cruzan con un cadena lineal, una cadena lineal que se cruza con una multilínea, varios puntos que se cruzan con un polígono y una cadena lineal que se cruza con un polígono. En tres de los cuatro casos, la geometría b se cruza con la geometría a. En el cuarto caso la geometría a es un multipunto que no cruza la línea pero sí que entra en contacto con el área interior del polígono de la geometría b.



Las geometrías en negro representan la geometría a; las geometrías en gris representan la geometría b.

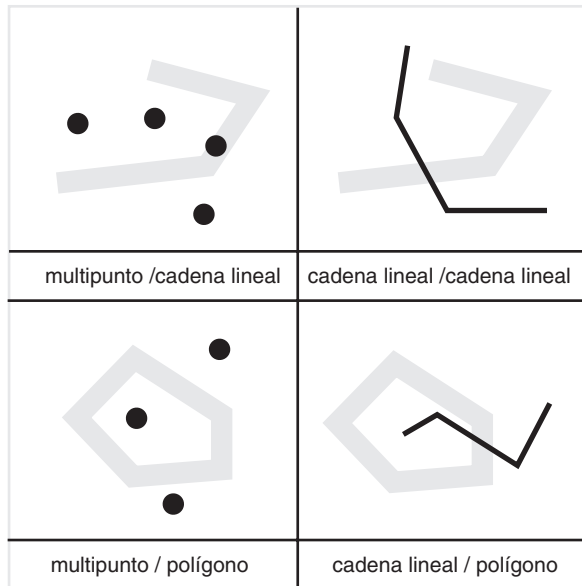


Figura 46. ST\_Crosses

La matriz patrón de la Tabla 48 es aplicable si la primera geometría es un punto o multipunto, o si la primera geometría es una curva o multicurva y la segunda geometría es una superficie. La matriz indica que los interiores deben formar intersección y que el interior de la primera geometría (a) debe formar intersección con el exterior de la segunda geometría (b).

Tabla 48. Matriz de ST\_Crosses (1)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	V	*	V
Exterior Geometría a	*	*	*

La matriz patrón de la Tabla 49 es aplicable si ambas geometrías son curvas o multicurvas. El 0 indica que la intersección de los interiores debe ser un punto (dimensión 0). Si la dimensión de esta intersección es 1 (intersección con una cadena lineal), la función ST\_Crosses devuelve un valor de 0 (que indica que las geometrías no se solapan); sin embargo, la función ST\_Overlaps devuelve un valor de 1 (que indica que las geometrías se solapan).

Tabla 49. Matriz de ST\_Crosses (2)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	0	*	*
Exterior Geometría a	*	*	*

## ST\_Overlaps

Utilice ST\_Overlaps para determinar si dos geometrías de la misma dimensión se solapan.

ST\_Overlaps compara dos geometrías de la misma dimensión. Esta función devuelve un 1 (uno) si el conjunto de intersección da como resultado una geometría que es diferente de las dos geometrías originales, pero que tiene la misma dimensión.

Las geometrías en negro representan la geometría *a*; las geometrías en gris representan la geometría *b*. En todos los casos, ambas geometrías tienen la misma dimensión y una se solapa parcialmente con la otra. El área de solapamiento es una nueva geometría, que tiene la misma dimensión que las geometrías *a* y *b*.

La siguiente figura ilustra el solapamiento de las geometrías. Los tres ejemplos muestran solapamientos con puntos, cadenas lineales y polígonos. Con puntos, los puntos en sí se solapan. Con cadenas lineales, una parte de la línea se solapa. Con polígonos, una parte del área se solapa.

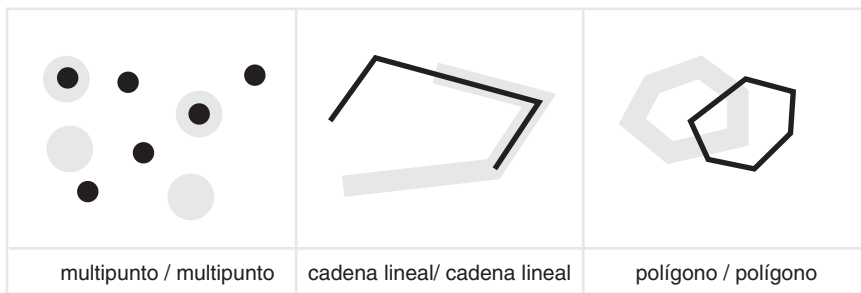


Figura 47. ST\_Overlaps

La matriz patrón de la Tabla 50 es aplicable si ambas geometrías son puntos, multipuntos, superficies o multisuperficies. ST\_Overlaps devuelve un 1 si el interior de cada geometría forma intersección con el interior y exterior de la otra geometría.

Tabla 50. Matriz de ST\_Overlaps (1)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	V	*	V
Exterior Geometría a	V	*	*

La matriz patrón de la Tabla 51 es aplicable si ambas geometrías son curvas o multicurvas. En este caso, la intersección de las geometrías debe dar lugar a una geometría cuya dimensión sea 1 (otra curva). Si la dimensión de la intersección de los interiores es 0, ST\_Overlaps devuelve un valor de 0 (que indica que las geometrías no se solapan); sin embargo, la función ST\_Crosses devuelve un valor de 1 (que indica que las geometrías se cruzan).

Tabla 51. Matriz de ST\_Overlaps (2)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	1	*	V
Exterior Geometría a	V	*	*

## ST\_Touches

ST\_Touches devuelve un valor de 1 (uno) si todos los puntos comunes a ambas geometrías sólo se pueden encontrar en los perímetros.

Los interiores de las geometrías no deben formar intersección entre sí. Como mínimo una de las geometrías debe ser una curva, superficie, multicurva o multisuperficie.

Las geometrías en negro representan la geometría a; las geometrías en gris representan la geometría b. En todos los casos, el perímetro de la geometría b forma intersección con la geometría a. El interior de la geometría b permanece separado de la geometría a.

La figura siguiente muestra ejemplos de contactos entre distintos tipos de geometría, como por ejemplo punto y cadena lineal, cadena lineal y multilínea, punto y polígono, y cadena lineal y polígono.

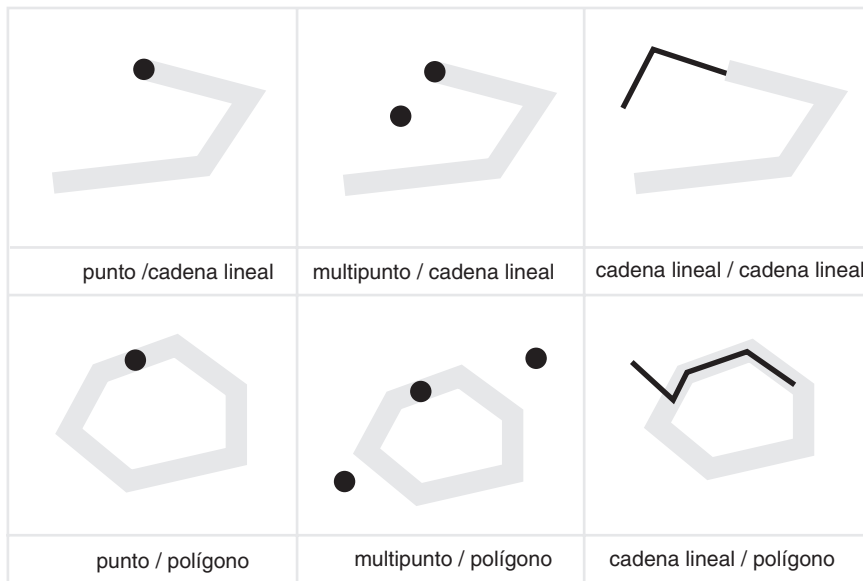


Figura 48. ST\_Touches

Las matrices patrón muestran que la función ST\_Touches devuelve un 1 (uno) cuando los interiores de la geometría no forman intersección, y el perímetro de cualquiera de las dos geometrías forma intersección con el interior o perímetro de la otra geometría.

Tabla 52. Matriz de ST\_Touches (1)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	F	V	*
Exterior Geometría a	*	*	*

Tabla 53. Matriz de ST\_Touches (2)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	V	*	*
Interior Geometría a	F	*	*
Exterior Geometría a	*	*	*

Tabla 54. Matriz de ST\_Touches (3)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	V	*
Interior Geometría a	F	*	*

Tabla 54. Matriz de *ST\_Touches* (3) (continuación)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Exterior Geometría a	*	*	*

## Funciones que comparan envolturas de geometrías

*ST\_EnvIntersects* y *ST\_MBRIntersects* son similares en cuanto que determinan si el rectángulo más pequeño que engloba a una geometría forma intersección con el rectángulo más pequeño que engloba a otra geometría. Hasta ahora, el término habitual utilizado para designar este rectángulo es el de envoltura. Los multipolígonos, polígonos, multilíneas y cadenas lineales curvadas se ajustan a los lados de sus envolturas; las cadenas lineales horizontales o verticales y los puntos son ligeramente menores que sus envolturas. *ST\_EnvIntersects* determina si las envolturas de las geometrías forman intersección.

Para designar el área rectangular más pequeña en la que puede estar contenida una geometría se utiliza el término "rectángulo delimitador mínimo" (minimum bounding rectangle, MBR). Las envolturas que rodean a los multipolígonos, polígonos, multilíneas y cadenas lineales curvadas son en realidad un MBR. En cambio, las envolturas que rodean a cadenas lineales horizontales o verticales y a los puntos no son un MBR, pues no constituyen un área mínima donde puedan caber esas geometrías. Esas geometrías no ocupan ningún espacio definible y por tanto no pueden tener un MBR. Pero, por convenio, se considera que estas geometrías constituyen sus propios MBR. Por tanto, con respecto a multipolígonos, polígonos, multilíneas y cadenas lineales curvadas, *ST\_MBRIntersects* verifica la condición de intersección para los mismos rectángulos delimitadores para los que lo hace *ST\_EnvIntersects*. Excepto para las cadenas lineales horizontales y verticales y los puntos, *ST\_MBRIntersects* verifica las intersecciones de las propias geometrías.

### **ST\_EnvIntersects**

*ST\_EnvIntersects* devuelve un 1 (uno) si las envolturas de dos geometrías forman intersección. Es una función auxiliar que implementa de forma eficaz *ST\_Intersects* (*ST\_Envelope*(g1), *ST\_Envelope*(g2)).

### **ST\_MBRIntersects**

*ST\_MBRIntersects* devuelve un 1 (uno) si los rectángulos delimitadores mínimos (los MBR) de dos geometrías forman intersección.

## Funciones que comprueban si dos elementos son idénticos

### **ST\_EqualCoordsys**

*ST\_EqualCoordsys* devuelve un valor de 1 (uno) si dos definiciones de sistemas de coordenadas son idénticas.

A la hora de comparación de las definiciones, *ST\_EqualCoordsys* no tiene en cuenta las diferencias referentes al uso de mayúsculas/minúsculas, espacios, paréntesis y la representación de números de coma flotante.

### **ST\_Equals**

*ST\_Equals* devuelve un valor de 1 (uno) si dos geometrías son idénticas.

El orden de los puntos utilizados para definir las geometrías no afecta a la comprobación de igualdad.

En los seis ejemplos (punto, multipunto, cadena lineal, multilínea, polígono y multipolígono) geometría a y geometría b son las mismas.




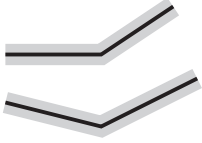
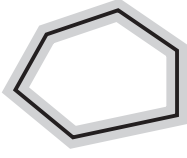
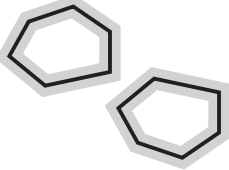
	
punto / punto	multipunto / multipunto
	
Cadena lineal /cadena lineal	multilínea /multilínea
	
polígono/ polígono	multipolígono /multipolígono

Figura 49. *ST\_Equals*. Las geometrías en negro representan la geometría a; las geometrías en gris representan la geometría b. En todos los casos, la geometría a equivale a la geometría b.

Tabla 55. *Matriz de igualdad*. La matriz patrón DE-9IM para comprobar la condición de igualdad verifica que los interiores forman intersección y que ningún interior o perímetro de cualquiera de las geometrías forma intersección con el exterior de la otra.

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	F
Interior Geometría a	V	*	F
Exterior Geometría a	F	F	*

## ST\_EqualSRS

Utilice *ST\_EqualSRS* para probar si dos sistemas de referencia espacial son idénticos.

*ST\_EqualSRS* devuelve un valor de 1 (uno) si dos sistemas de referencia espacial son idénticos, a condición de que el identificador numérico de uno o ambos de los sistemas sea nulo.

## Función que comprueba si no hay intersección entre dos geometrías

ST\_Disjoint devuelve un valor de 1 (uno) si la intersección de las dos geometrías es un conjunto vacío. El resultado devuelto por esta función es exactamente el opuesto al de ST\_Intersects.

La ilustración muestra distintas geometrías y cómo los perímetros no forman intersección en ningún punto.

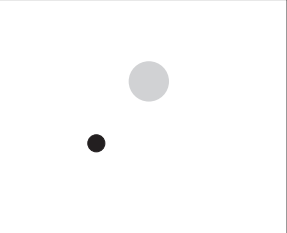
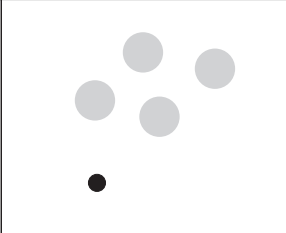
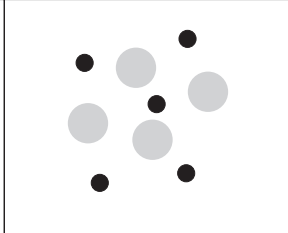
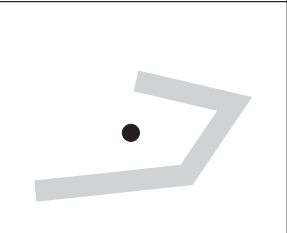
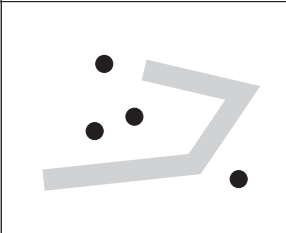
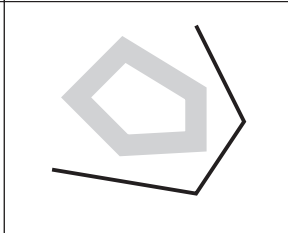
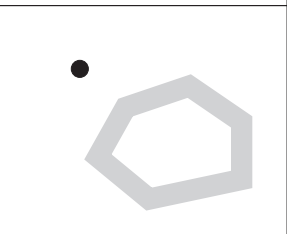

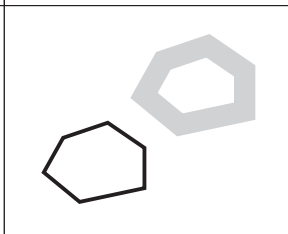
		
punto / punto	punto / multipunto	multipunto / multipunto
		
punto / cadena lineal	multilínea / multilínea	polígono / cadena lineal
		
punto / polígono	multipunto / multipolígono	polígono / polígono

Figura 50. ST\_Disjoint. Las geometrías en negro representan la geometría a; las geometrías en gris representan la geometría b. En todos los casos, la geometría a y la geometría b están inconexas la una respecto a la otra.

Tabla 56. Matriz de ST\_Disjoint. Esta matriz indica que ni los interiores ni los perímetros de ninguna de las dos geometrías forman intersección.

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	F	F	*
Interior Geometría a	F	F	*
Exterior Geometría a	*	*	*

---

## Función que compara geometrías con la serie de la matriz patrón DE-9IM

La función ST\_Relate compara dos geometrías y devuelve un valor de 1 (uno) si las geometrías satisfacen las condiciones especificadas por la serie de la matriz patrón DE-9IM; en caso contrario, la función devuelve un valor de 0 (cero).

---

## Funciones que devuelven información sobre propiedades de geometrías

Esta sección describe las funciones espaciales que devuelven información acerca de las propiedades de las geometrías. Esta información comprende:

- Tipos de datos de las geometrías
- Coordenadas y medidas dentro de una geometría
- Anillos, perímetros, envolturas y rectángulos delimitadores mínimos (los MBR)
- Dimensiones
- Indicación de si la geometría es cerrada, vacía o simple
- Geometrías base que componen una colección de geometrías
- Sistemas de referencia espacial

Algunas propiedades son geometrías por sí mismas; por ejemplo, los anillos exterior e interior de una superficie o los puntos inicial y final de una curva. Estas geometrías son producidas por algunas de las funciones incluidas en esta categoría. Las funciones que producen otras clases de geometrías como, por ejemplo, las geometrías que representan zonas que circundan a un lugar determinado, pertenecen a otra categoría. Para obtener información sobre esta otra categoría, denominada "Funciones espaciales que generan nuevas geometrías", consulte el enlace o referencia cruzada apropiados al final de esta sección.

---

## Función que devuelve información sobre el tipo de datos

ST\_GeometryType toma una geometría como parámetro de entrada y devuelve el nombre de tipo calificado al completo del tipo dinámico de dicha geometría.

---

## Funciones que devuelven información sobre coordenadas y medidas

Las funciones siguientes devuelven información sobre las coordenadas y medidas de una geometría. Por ejemplo, ST\_X devuelve la coordenada X de un punto especificado, ST\_MaxX devuelve la coordenada X mayor de una geometría y ST\_MinX devuelve la coordenada X menor de una geometría.

Estas funciones son:

- ST\_CoordDim
- ST\_IsMeasured
- ST\_IsValid
- ST\_Is3D
- ST\_M
- ST\_MaxM
- ST\_MaxX

- ST\_MaxY
- ST\_MaxZ
- ST\_MinM
- ST\_MinX
- ST\_MinY
- ST\_MinZ
- ST\_X
- ST\_Y
- ST\_Z

## ST\_CoordDim

ST\_CoordDim devuelve un valor que indica qué tipos de coordenadas tiene una geometría y si la geometría también contiene alguna medida.

Este valor se denomina *dimensión de coordenada*. Una dimensión de coordenada no es lo mismo que la propiedad denominada dimensión. Esta última indica si una geometría tiene un ancho o longitud, no si contiene coordenadas de un tipo específico o medidas.

## ST\_IsMeasured

ST\_IsMeasured toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría tiene coordenadas M (medidas). En caso contrario, se devuelve un 0 (cero).

## ST\_IsValid

ST\_IsValid toma una geometría como parámetro de entrada y devuelve un 1 si la geometría es válida. En caso contrario, se devuelve un 0 (cero). Una geometría es válida sólo si todos los atributos contenidos en el tipo estructurado son coherentes con la representación interna de los datos de la geometría, y si la representación interna no está dañada.

## ST\_Is3D

ST\_Is3d toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría tiene coordenadas Z. En caso contrario, se devuelve un 0 (cero).

## ST\_M

Si un punto determinado contiene una medida, ST\_M puede tomar el punto como parámetro de entrada y obtener la medida.

## ST\_MaxM

ST\_MaxM toma una geometría como parámetro de entrada y devuelve su medida máxima.

## ST\_MaxX

ST\_MaxX toma una geometría como parámetro de entrada y devuelve su coordenada X máxima.

## ST\_MaxY

ST\_MaxY toma una geometría como parámetro de entrada y devuelve su coordenada Y máxima.



## **ST\_MaxZ**

ST\_MaxZ toma una geometría como parámetro de entrada y devuelve su coordenada Z máxima.

## **ST\_MinM**

ST\_MinM toma una geometría como parámetro de entrada y devuelve su medida mínima.

## **ST\_MinX**

ST\_MinX toma una geometría como parámetro de entrada y devuelve la coordenada X mínima.

## **ST\_MinY**

ST\_MinY toma una geometría como parámetro de entrada y devuelve su coordenada Y mínima.

## **ST\_MinZ**

ST\_MinZ toma una geometría como parámetro de entrada y devuelve su coordenada Z mínima.

## **ST\_X**

ST\_X toma un punto como parámetro de entrada y devuelve la coordenada X del punto.

## **ST\_Y**

ST\_Y toma un punto como parámetro de entrada y devuelve la coordenada Y del punto.

## **ST\_Z**

Si un punto determinado contiene una coordenada Z, ST\_Z puede tomar el punto como parámetro de entrada y obtener la coordenada Z.

---

## **Funciones que devuelven información sobre geometrías contenidas en una geometría**

Las funciones siguientes devuelven información sobre geometrías contenidas en una geometría. Algunas funciones identifican puntos determinados contenidos en una geometría; otras obtienen el número de geometrías contenidas en una colección.

Estas funciones son:

- ST\_Centroid
- ST\_EndPoint
- ST\_GeometryN
- ST\_LineStringN
- ST\_MidPoint
- ST\_NumGeometries
- ST\_NumLineStrings
- ST\_NumPoints

- ST\_NumPolygons
- ST\_PointN
- ST\_PolygonN
- ST\_StartPoint

## ST\_Centroid

ST\_Centroid toma una geometría como parámetro de entrada y devuelve el centro geográfico (el centro del rectángulo delimitador mínimo de la geometría) en forma de punto.

## ST\_EndPoint

ST\_Endpoint toma una curva como parámetro de entrada y devuelve el punto que es el último punto de la curva.

## ST\_GeometryN

ST\_GeometryN toma una colección de geometrías y un índice como parámetros de entrada y devuelve la geometría de la colección que está identificada por el índice.

## ST\_LineStringN

ST\_LineStringN toma un multilínea y un índice como parámetros de entrada y devuelve la cadena lineal identificada por el índice.

## ST\_MidPoint

ST\_MidPoint toma una curva como parámetro de entrada y devuelve el punto en la curva que es equidistante de los dos puntos de la curva, medido a lo largo de la curva.

## ST\_NumGeometries

ST\_NumGeometries toma una colección de geometrías como parámetro de entrada y devuelve el número de geometrías de la colección.

## ST\_NumLineStrings

ST\_NumLineStrings toma una multilínea como parámetro de entrada y devuelve el número de cadenas lineales de la multilínea.

## ST\_NumPoints

ST\_NumPoints toma una geometría como parámetro de entrada y devuelve el número de puntos que se utilizaron para definir esa geometría. Por ejemplo, si la geometría es un polígono y se utilizaron cinco puntos para definir ese polígono, el número devuelto es 5.

## ST\_NumPolygons

ST\_NumPolygons toma un multipolígono como parámetro de entrada y devuelve el número de polígonos que contiene.

## ST\_PointN

ST\_PointN toma una cadena lineal o un multipunto, y un índice como parámetros de entrada y devuelve el punto identificado por el índice y perteneciente a la cadena lineal o multipunto.

## ST\_PolygonN

ST\_PolygonN toma un multipolígono y un índice como parámetros de entrada y devuelve el polígono identificado por el índice.

## ST\_StartPoint

ST\_StartPoint toma una curva como parámetro de entrada y devuelve el primer punto de la curva.

---

## Funciones que proporcionan información sobre perímetros, envolturas y anillos

Las funciones siguientes devuelven información sobre demarcaciones que separan una parte interior de una geometría con respecto a una parte exterior, o que separan a la propia geometría con respecto al espacio que externo que la rodea. Por ejemplo, ST\_Boundary devuelve el perímetro de una geometría representado en forma de curva.

Estas funciones son:

- ST\_Boundary
- ST\_Envelope
- ST\_EnvIntersects
- ST\_ExteriorRing
- ST\_InteriorRingN
- ST\_MBR
- ST\_MBRIntersects
- ST\_NumInteriorRing
- ST\_Perimeter

## ST\_Envelope

ST\_Envelope toma una geometría como parámetro de entrada y devuelve la envoltura de la geometría. La envoltura es un rectángulo que está representado por un polígono.

## ST\_EnvIntersects

ST\_EnvIntersects toma dos geometrías como parámetros de entrada y devuelve un 1 si las envolturas de las dos geometrías forman intersección. En caso contrario, se devuelve un 0 (cero).

## ST\_ExteriorRing

ST\_ExteriorRing toma un polígono como parámetro de entrada y devuelve su anillo exterior en forma de curva.

## ST\_InteriorRingN

ST\_InteriorRingN toma un polígono y un índice como parámetros de entrada y devuelve el anillo interior identificado por dicho índice en forma de cadena lineal. Los anillos interiores se organizan según las normas definidas por las rutinas definidas por las rutinas de verificación de la geometría interna.

## ST\_MBR

ST\_MBR toma una geometría como parámetro de entrada y devuelve el rectángulo delimitador mínimo de la misma.

## ST\_MBRIntersects

ST\_MBRIntersects devuelve un 1 (uno) si los rectángulos delimitadores mínimos (los MBR) de dos geometrías forman intersección.

## ST\_NumInteriorRing

ST\_NumInteriorRing toma un polígono como parámetro de entrada y devuelve el número de sus anillos interiores.

## ST\_Perimeter

ST\_Perimeter toma como parámetros de entrada una superficie o multisuperficie y, opcionalmente, una unidad de medida y devuelve el perímetro de la superficie o multisuperficie (es decir, la longitud de sus límites) expresado en la unidad de medida proporcionada.

---

## Funciones que devuelven información sobre las dimensiones de una geometría

Las funciones siguientes devuelven información sobre la dimensión de una geometría. Por ejemplo, ST\_Area proporciona el área cubierta por una geometría determinada.

Estas funciones son:

- ST\_Area
- ST\_Dimension
- ST\_Length

### ST\_Area

ST\_Area toma como parámetros de entrada una geometría y, opcionalmente, una unidad de medida y devuelve el área cubierta por la geometría, expresada en la unidad de medida proporcionada.

### ST\_Dimension

ST\_Dimension toma una geometría como parámetro de entrada y devuelve la dimensión del mismo.

### ST\_Length

ST\_Length toma como parámetros de entrada una curva o multicurva y, opcionalmente, una unidad de medida y devuelve la longitud de la curva o multicurva proporcionada, expresada en la unidad de medida proporcionada.

---

## Funciones que indican si una geometría es cerrada, vacía o simple

Las funciones siguientes indican si:

- Si una curva o multicurva determinada es cerrada (es decir, si el punto inicial y final de la curva o multicurva es el mismo punto)
- Si una geometría determinada está vacía (es decir, carece de puntos)
- Si una curva, multicurva o multipunto es simple (es decir, si tales geometrías tienen configuraciones típicas)

### **ST\_IsClosed**

ST\_IsClosed toma una curva o multicurva como parámetro de entrada y devuelve un 1 si dicha curva o multicurva es cerrada. En caso contrario, se devuelve un 0 (cero).

### **ST\_IsEmpty**

ST\_IsEmpty toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría está vacía. En caso contrario, se devuelve un 0 (cero).

### **ST\_IsSimple**

ST\_IsSimple toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría es simple. En caso contrario, se devuelve un 0 (cero).

## **Funciones que identifican el sistema de referencia espacial de una geometría**

Las funciones siguientes devuelven valores que identifican el sistema de referencia espacial correspondiente a una geometría. Además, la función ST\_SrsID puede cambiar el sistema de referencia espacial de una geometría sin cambiar ni transformar la geometría.

### **ST\_SrsId (también llamada ST\_SRID)**

ST\_SrsId (o ST\_SRID) toma como parámetros de entrada una geometría y, opcionalmente, un identificador de sistemas de referencia espacial. Los datos que devuelve esta función dependen de los parámetros de entrada que se hayan especificado:

- Si se especifica el identificador de sistemas de referencia espacial, la función devuelve la geometría con su identificador cambiado al identificador especificado. No se realiza ninguna transformación de la geometría.
- Si no se proporciona como parámetro de entrada ningún identificador de sistemas de referencia espacial, se devuelve el identificador actual de la geometría especificada.

### **ST\_SrsName**

ST\_SrsName toma una geometría como parámetro de entrada y devuelve el nombre del sistema de referencia espacial en el que está representada dicha geometría.

## **Funciones que generan nuevas geometrías a partir de las existentes**

Esta sección describe la categoría de funciones que obtienen nuevas geometrías a partir de otras existentes. Esta categoría no incluye las funciones que obtienen geometrías representativas de propiedades de otras geometrías. Sino que estas funciones realizan estas acciones:

- Convierten geometrías en otras geometrías
- Crean geometrías que representan configuraciones de espacio
- Obtienen geometrías individuales a partir de varias geometrías
- Crean geometrías basándose en medidas
- Crean modificaciones de geometrías

---

## Funciones que convierten una geometría en otra

Las funciones siguientes pueden convertir geometrías pertenecientes a un supertipo en las geometrías correspondientes de un subtipo. Por ejemplo, la función `ST_ToLineString` puede convertir una cadena lineal de tipo `ST_Geometry` en una cadena lineal de tipo `ST_LineString`. Algunas de estas funciones pueden también combinar geometrías base y colecciones de geometrías para formar una colección de geometrías individual. Por ejemplo, `ST_ToMultiLine` puede convertir una cadena lineal y una multilínea en una multilínea individual.

### **ST\_Polygon**

`ST_Polygon` puede crear un polígono a partir de una cadena lineal cerrada. La cadena lineal definirá el anillo exterior del polígono.

### **ST\_ToGeomColl**

`ST_ToGeomColl` toma una geometría como parámetro de entrada y la convierte a un conjunto de geometrías.

### **ST\_ToLineString**

`ST_ToLineString` toma una geometría como parámetro de entrada y la convierte en una cadena lineal.

### **ST\_ToMultiLine**

`ST_ToMultiLine` toma una geometría como parámetro de entrada y la convierte en una multilínea.

### **ST\_ToMultiPoint**

`ST_ToMultiPoint` toma una geometría como parámetro de entrada y la convierte en multipunto.

### **ST\_ToMultiPolygon**

`ST_ToMultiPolygon` toma una geometría como parámetro de entrada y la convierte en un multipolígono.

### **ST\_ToPoint**

`ST_ToPoint` toma una geometría como parámetro de entrada y la convierte en un punto.

### **ST\_ToPolygon**

`ST_ToPolygon` toma una geometría como parámetro de entrada y la convierte en un polígono.

---

## Funciones que crean nuevas geometrías con configuraciones espaciales diferentes

Utilizando geometrías existentes como punto de partida, las funciones siguientes crean nuevas geometrías que representan áreas circulares u otras configuraciones de espacio. Por ejemplo, dado un punto que representa el centro de un aeropuerto proyectado, `ST_Buffer` puede crear una superficie que representa, en formato circular, la extensión propuesta del aeropuerto.

Estas funciones son:

- `ST_Buffer`
- `ST_ConvexHull`
- `ST_Difference`
- `ST_Intersection`
- `ST_SymDifference`

### ST\_Buffer

La función `ST_Buffer` puede generar una geometría nueva que sobresale de una geometría existente con un radio especificado. La nueva geometría será una superficie cuando se cree una zona de separación en torno a la geometría existente o cuando los elementos de una colección están tan cerca unos de otros, que existe solapamiento de las zonas de separación que rodean a los elementos individuales de la colección. Sin embargo, cuando las zonas de separación están separadas, se originan superficies de separación individuales, en cuyo caso `ST_Buffer` devuelve una multisuperficie.

La figura siguiente ilustra la zona de separación situada alrededor de elementos únicos y solapados.

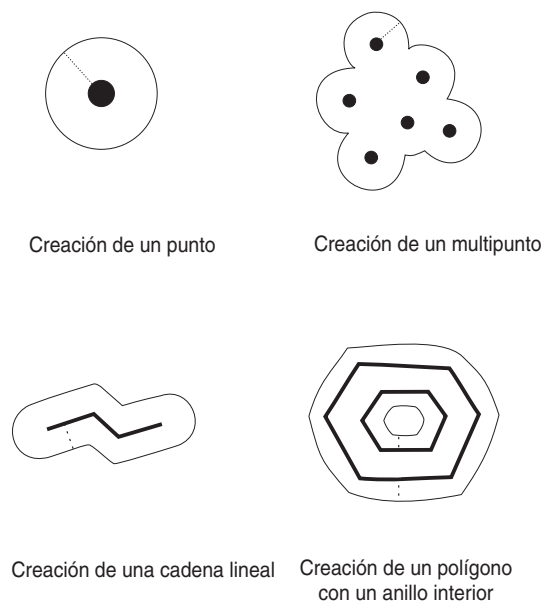


Figura 51. `ST_Buffer`

La función `ST_Buffer` acepta tanto una distancia positiva como negativa, pero sólo las geometrías cuya dimensión sean 2 (superficies y multisuperficies) aplican una

zona de separación negativa. Se utiliza el valor absoluto de la distancia de separación cuando la dimensión de la geometría original es menor que 2 (todas las geometrías que no sean superficies ni multisuperficies).

En general, para los anillos exteriores, las distancias de separación positivas generan anillos de superficie que están separados del centro de la geometría original; las distancias de separación negativas generan anillos de superficie o multisuperficie dirigidos hacia el centro. Para los anillos interiores de una superficie o multisuperficie, una distancia de separación positiva genera un anillo de separación dirigido hacia el centro, y una distancia de separación negativa genera un anillo de separación separado del centro.

El proceso de creación de zonas de separación produce la fusión de las superficies que se solapan. Las distancias negativas mayores que la mitad del ancho interior máximo de un polígono dan lugar a una geometría vacía.

## ST\_ConvexHull

La función ST\_ConvexHull devuelve la cáscara convexa de cualquier geometría que tenga como mínimo tres vértices formando una superficie convexa. Los vértices son pares de coordenadas X e Y situados dentro de las geometrías. Una cáscara convexa es el polígono convexo más pequeño que puede ser formado por todos los vértices de un conjunto determinado de vértices.

La siguiente ilustración muestra cuatro ejemplos de cáscara convexa. En el primer ejemplo se ha dibujado una forma irregular parecida a la letra c. La c está delimitada por la cáscara convexa. En el cuarto ejemplo hay cuatro puntos con líneas que siguen un patrón en forma de zigzag. La línea convexa discurre entre los puntos cuatro y dos en un lado y el tres y el uno en el otro lado.

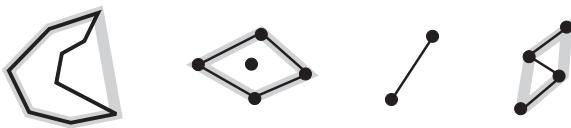


Figura 52. ST\_ConvexHull

## ST\_Difference

ST\_Difference toma como entrada dos geometrías de la misma dimensión. La función ST\_Difference devuelve la parte de la primera geometría que no forma intersección con la segunda. Esta operación es el equivalente espacial de la operación lógica AND NOT. La parte de geometría devuelta por ST\_Difference constituye por sí misma una geometría: una colección que tiene la misma dimensión que las geometrías utilizadas como entrada. Si estas dos geometrías son iguales, es decir, si ocupan el mismo espacio, la geometría que se devuelve está vacía.

A la izquierda de cada flecha hay dos geometrías que se proporcionan a ST\_Difference como entrada. A la derecha de cada fila es la salida de ST\_Difference. Si una parte de la primera geometría forma intersección con la segunda, el resultado de la función es esa parte de la primera geometría que no forma intersección. Si las geometrías proporcionadas como entrada son iguales, la salida resultante será una geometría vacía (indicada por el término nil)

Esta figura muestra la entrada y la salida para ST\_Difference. Por ejemplo, si la entrada son puntos, y el punto a y el punto b son iguales, la salida será un valor



nulo. Si el punto a y el punto b son diferentes, la salida sería un nuevo punto situado entre los dos. Si la entrada fuera un polígono para b y un polígono más pequeño aunque idéntico para la geometría a situada dentro del primero, el resultado sería un valor nulo. Si los polígonos se solaparan, la salida se correspondería con los bordes exteriores de los polígonos combinados.

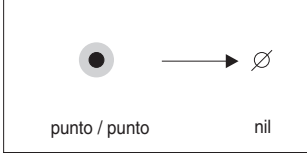
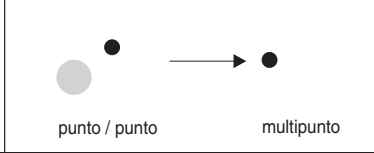
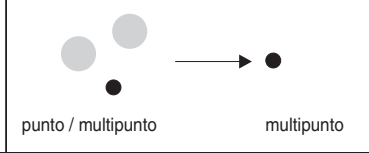
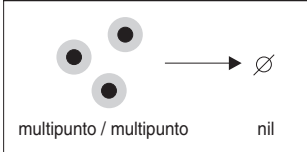
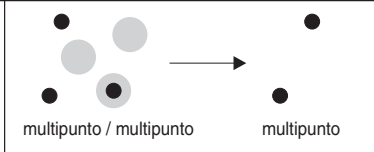
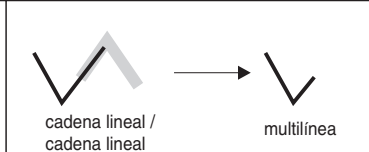
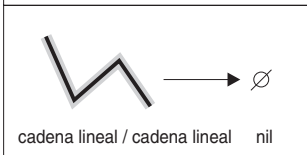
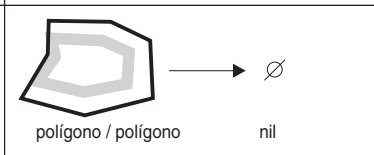
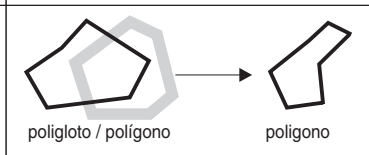
 punto / punto → ∅ nil	 punto / punto → multipunto	 punto / multipunto → multipunto
 multipunto / multipunto → ∅ nil	 multipunto / multipunto → multipunto	 cadena lineal / cadena lineal → multilinea
 cadena lineal / cadena lineal → ∅ nil	 poligono / poligono → ∅ nil	 poligloto / polígono → poligono

Figura 53. ST\_Difference

## ST\_Intersection

La función ST\_Intersection devuelve un conjunto de puntos, representados en forma de geometría, que definen la intersección de dos geometrías proporcionadas.

Si las geometrías proporcionadas a ST\_Intersection como entrada no forman intersección o bien si forman intersección y la dimensión y de la intersección es menor que las dimensiones de las geometrías, ST\_Intersection devuelve una geometría vacía.

A la izquierda de cada flecha hay dos geometrías que forman intersección que se suministran a ST\_Intersection como entrada. A la derecha de cada flecha está la salida de ST\_Intersection: una geometría que representa la intersección creada por las geometrías de la izquierda.

Esta figura ilustra diez ejemplos de salida de ST\_Intersection, que devuelve información sobre dónde forman intersección las geometrías suministradas. Por ejemplo, si b fuera una cadena lineal y la geometría a fuera un punto de la línea, la salida sería el multipunto en el que convergen la geometría a y la geometría b. Si la geometría a y la geometría b fueran polígonos solapados, la salida sería un nuevo multipolígono sólo con la porción solapada.

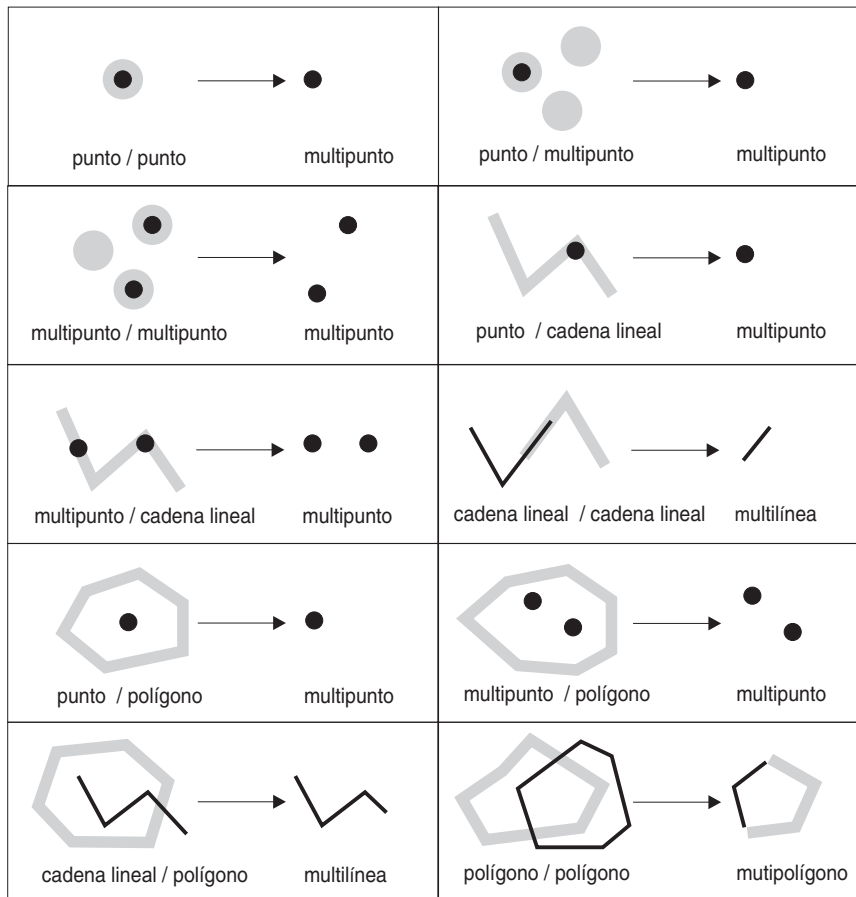


Figura 54. *ST\_Intersection*

## ST\_SymDifference

La función *ST\_SymDifference* devuelve la diferencia simétrica (el equivalente espacial de la operación XOR lógica) de dos geometrías que forman intersección y tienen la misma dimensión. Si estas geometrías son iguales, *ST\_SymDifference* devuelve una geometría vacía. Si las geometrías no son iguales, entonces una parte de una o ambas geometrías quedará fuera del área de intersección.

---

## Funciones que obtienen una geometría a partir de varias

Las funciones siguientes obtienen geometrías individuales a partir de varias geometrías. Por ejemplo, *ST\_Union* combina dos geometrías y forma una geometría individual.

### MBR Aggregate (Agregado MBR)

El uso combinado de las funciones *ST\_BuildMBRAggr* y *ST\_GetAggrResult* agrega una columna de geometrías de una columna seleccionada a una geometría individual creando un rectángulo que representa el rectángulo delimitador mínimo (minimum bounding rectangle, MBR) que encierra todas las geometrías contenidas en la columna. Las coordenadas Z y M se descartan cuando se calcula el agregado.

## ST\_Union

La función ST\_Union devuelve el conjunto de unión de dos geometrías.

Esta operación es el equivalente espacial de la operación lógica OR. Las dos geometrías deben tener la misma dimensión. ST\_Union siempre devuelve el resultado en forma de colección.

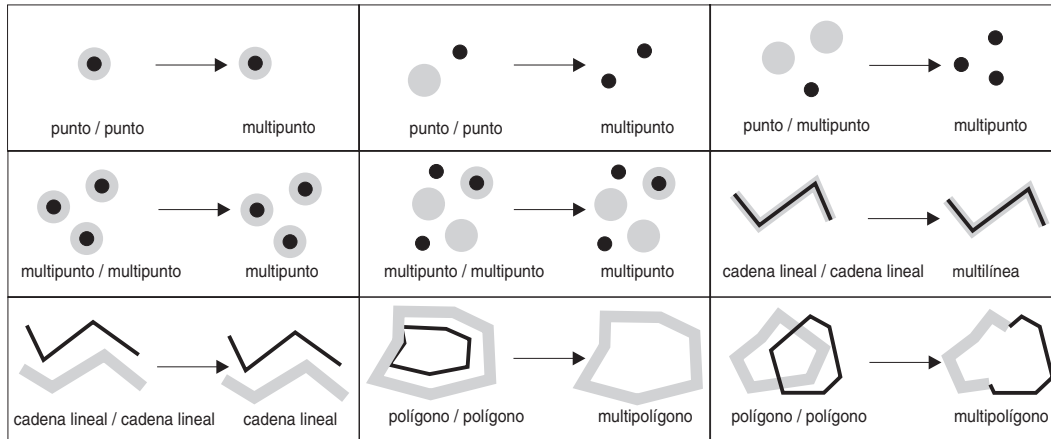


Figura 55. ST\_Union

## Union Aggregate (Agregado de unión)

Un agregado de unión es la combinación de las funciones ST\_BuildUnionAggr y ST\_GetAggrResult. Esta combinación agrega una columna de geometrías de una tabla a una geometría individual mediante la creación de la unión.

## Funciones que trabajan con medidas

Las funciones que se describen en esta sección trabajan con geometrías que tienen valores de medida, y devuelven ya sea una nueva geometría basada en medias, ya sea la distancia con respecto a una ubicación sobre la geometría.

Estas funciones son:

### ST\_DistanceToPoint

ST\_DistanceToPoint toma una geometría de curva o multicurva y una geometría de punto como parámetros de entrada y devuelve la distancia junto con la geometría de curva en el punto especificado.

También se puede invocar esta función como método.

### Sintaxis

```
►►db2gse.ST_DistanceToPoint(—geometría_curva—,—geometría_punto—)►►
```

### Parámetro

#### geometría\_curva

Valor de tipo ST\_Curve o ST\_MultiCurve o uno de sus subtipos que representa la geometría que debe procesarse.

### geometría\_punto

Valor de tipo ST\_Point que representa un punto en la curva especificada.

### Tipo de retorno

DOUBLE

### Ejemplo

#### Ejemplo 1

La sentencia de SQL siguiente crea la tabla SAMPLE\_GEOMETRIES con dos columnas. La columna de ID identifica exclusivamente cada fila. La columna GEOMETRY ST\_LineString almacena geometrías de ejemplo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

La sentencia de SQL siguiente inserta dos filas en la tabla SAMPLE\_GEOMETRIES.

```
INSERT INTO sample_geometries(id, geometry)
VALUES
(1,ST_LineString('LINESTRING ZM(0 0 0 0, 10 100 1000 10000)',1)),
(2,ST_LineString('LINESTRING ZM(10 100 1000 10000, 0 0 0 0)',1))
```

La sentencia SELECT siguiente y el conjunto de resultados correspondiente muestra cómo usar la función ST\_DistanceToPoint para buscar la distancia respecto al punto en la ubicación (1.5, 15.0).

```
SELECT ID, DECIMAL(ST_DistanceToPoint(geometry,ST_Point(1.5,15.0,1)),10,5)
AS DISTANCE FROM sample_geometries
```

ID	DISTANCE
1	15.07481
2	85.42394

2 registro(s) seleccionados.

## ST\_FindMeasure o ST\_LocateAlong

ST\_FindMeasure o ST\_LocateAlong toma una geometría y una medida como parámetros de entrada y devuelve un multipunto o una multicurva que forma parte de la geometría proporcionada y cuya medida es exactamente la especificada.

Para los puntos y multipuntos, el resultado es todos los puntos que tienen la medida especificada. En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. Para las superficies y multisuperficies, el cálculo se realiza sobre el perímetro de la geometría.

Para los puntos y multipuntos, si no se encuentra la medida proporcionada, el resultado es una geometría vacía. Para todas las demás geometrías, si la medida proporcionada es menor que la medida más pequeña de la geometría o mayor que la medida más alta de la geometría, el resultado es una geometría vacía. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

► db2gse.ST\_FindMeasure (—geometría—, —medida—) ►  
db2gse.ST\_LocateAlong

## Parámetro

### geometría

Un valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría en la que se buscan partes cuyas coordenadas M (medidas) contienen *medida*.

### medida

Un valor de tipo DOUBLE que es la medida que las partes de *geometría* deben incluirse en el resultado.

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplos

### Ejemplo 1

La sentencia CREATE TABLE siguiente crea la tabla SAMPLE\_GEOMETRIES. SAMPLE\_GEOMETRIES tiene dos columnas: la columna ID, que sirve para identificar cada columna de forma unívoca, y la columna GEOMETRY de tipo ST\_Geometry, que contiene la geometría de ejemplo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_geometries(id SMALLINT, geometry ST_GEOMETRY)
```

Las sentencias INSERT siguientes insertan dos filas. La primera es una cadena lineal; la segunda es un multipunto.

```
INSERT INTO sample_geometries(id, geometry)
VALUES
  (1, ST_LineString('linestring m (2 2 3, 3 5 3, 3 3 6, 4 4 8)', 1)),
  (2, ST_MultiPoint('multipoint m
  (2 2 3, 3 5 3, 3 3 6, 4 4 6, 5 5 6, 6 6 8)', 1))
```

### Ejemplo 2

En la sentencia SELECT siguiente y en el resultado correspondiente, la función ST\_FindMeasure busca puntos cuya medida es 7. El resultado obtenido para la primera fila es un punto. En cambio, el resultado obtenido para la segunda fila es un punto vacío. En el caso de elementos geográficos lineales (geometrías con una dimensión mayor que 0), ST\_FindMeasure puede interpolar el punto; en cambio, para multipuntos, la medida resultante debe coincidir exactamente.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 7))
  AS varchar(45)) AS measure_7
FROM sample_geometries
```

Resultados:

```
ID      MEASURE_7
-----
1 POINT M ( 3.50000000 3.50000000 7.00000000)
2 POINT EMPTY
```

### Ejemplo 3

En la sentencia SELECT siguiente y en el resultado correspondiente, la función ST\_FindMeasure devuelve un punto y un multipunto. La medida resultante (6) coincide con las medidas existentes en las fuentes de datos de ST\_FindMeasure y del multipunto.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 6))
              AS varchar(120)) AS measure_6
FROM sample_geometries
```

Resultados:

ID	MEASURE_6
1	POINT M ( 3.00000000 3.00000000 6.00000000)
2	MULTIPOINT M ( 3.00000000 3.00000000 6.00000000, 4.00000000 4.00000000 6.00000000, 5.00000000 5.00000000 6.00000000)

## ST\_MeasureBetween o ST\_LocateBetween

ST\_MeasureBetween o ST\_LocateBetween toma como parámetros de entrada una geometría y dos coordenadas M (medidas) y devuelve la parte de la geometría proporcionada que representa el conjunto de puntos desconectados situados entre las dos coordenadas M.

En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si la geometría proporcionada es una superficie o multisuperficie, ST\_MeasureBetween o ST\_LocateBetween se aplica a los anillos exterior e interior de la geometría. Si ninguna de las partes de la geometría proporcionada está dentro del intervalo definido por las coordenadas M proporcionadas, el resultado es una geometría vacía. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

Si la geometría resultante no está vacía, se devuelve un tipo multipunto o multilínea.

Ambas funciones también se pueden invocar como métodos.

### Sintaxis

```
db2gse.ST_MeasureBetween
db2gse.ST_LocateBetween

(—geometría—, —medidaInicial—, —medidaFinal—)
```

### Parámetros

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría donde se deben localizar las partes cuyos valores de medida están comprendidos entre *medidaInicial* y *medidaFinal*.

**medidaInicial**

Valor de tipo DOUBLE que representa el límite inferior del intervalo de medidas. Si este valor es nulo, no se aplica ningún límite inferior.

**medidaFinal**

Valor de tipo DOUBLE que representa el límite superior del intervalo de medidas. Si este valor es nulo, no se aplica ningún límite superior.

**Tipo de retorno**

db2gse.ST\_Geometry

**Ejemplo**

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

La coordenada M (medida) de una geometría está definida por el usuario. Es muy versátil porque puede representar cualquier entidad que desee medir; por ejemplo, una distancia de carretera, o medidas de temperatura, presión o pH.

Este ejemplo muestra el uso de la coordenada M para registrar datos recogidos de mediciones de pH. Un investigador obtiene la medida del pH del suelo en lugares determinados de una carretera. De acuerdo con sus técnicas habituales de trabajo, anota los valores necesarios en cada lugar donde toma una muestra del suelo: las coordenadas X e Y de ese lugar y el pH medido.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_LineString)
```

```
INSERT INTO sample_lines
VALUES (1, ST_LineString ('linestring m (2 2 3, 3 5 3,
                               3 3 6, 4 4 6,
                               5 5 6, 6 6 8)', 1 ) )
```

Para encontrar el tramo donde la acidez del suelo oscila entre 4 y 6, el investigador utilizaría esta sentencia SELECT:

```
SELECT id, CAST( ST_AsText( ST_MeasureBetween( 4, 6 )
AS VARCHAR(150) ) MEAS_BETWEEN_4_AND_6
FROM sample_lines
```

Resultados:

```
ID          MEAS_BETWEEN_4_AND_6
-----
1  LINESTRING M (3.00000000 4.33333300 4.00000000,
                 3.00000000 3.00000000 6.00000000,
                 4.00000000 4.00000000 6.00000000,
                 5.00000000 5.00000000 6.00000000)
```

**ST\_PointAtDistance**

ST\_PointAtDistance toma una geometría de curva o multicurva y una distancia como parámetros de entrada y devuelve la geometría de punto en la distancia dada sobre la geometría de curva.

También se puede invocar esta función como método.

## Sintaxis

► db2gse.ST\_PointAtDistance(—geometría—,—distancia—) ◀

## Parámetro

### geometría

Valor de tipo ST\_Curve o ST\_MultiCurve o uno de sus subtipos que representa la geometría que debe procesarse.

### distancia

Valor de tipo DOUBLE que representa la distancia sobre la geometría para localizar el punto.

## Tipo de retorno

db2gse.ST\_Point

## Ejemplo

### Ejemplo 1

La sentencia de SQL siguiente crea la tabla SAMPLE\_GEOMETRIES con dos columnas. La columna de ID identifica exclusivamente cada fila. La columna GEOMETRY ST\_LineString almacena geometrías de ejemplo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

La sentencia de SQL siguiente inserta dos filas en la tabla SAMPLE\_GEOMETRIES.

```
INSERT INTO sample_geometries(id, geometry)
VALUES
(1,ST_LineString('LINESTRING ZM(0 0 0 0, 10 100 1000 10000)',1)),
(2,ST_LineString('LINESTRING ZM(10 100 1000 10000, 0 0 0 0)',1))
```

La sentencia SELECT siguiente y el conjunto de resultados correspondiente muestra cómo usar la función ST\_PointAtDistance para buscar puntos a una distancia de 15 unidades de coordenada a partir del principio de la cadena lineal.

```
SELECT ID, VARCHAR(ST_AsText(ST_PointAtDistance(geometry, 15)), 50) AS POINTAT
FROM sample_geometries
```

```
ID          POINTAT
-----
1 POINT ZM(1.492556 14.925558 149 1493)
2 POINT ZM(8.507444 85.074442 851 8507)
```

2 registro(s) seleccionados.

---

## Funciones que crean formas modificadas de geometrías existentes

Las funciones siguientes crean formas modificadas de geometrías existentes. Por ejemplo, la función ST\_AppendPoint crea versiones ampliadas de curvas existentes. Cada versión contiene los puntos de una curva existente más un punto adicional.

Estas funciones son:

- ST\_AppendPoint
- ST\_ChangePoint
- ST\_Generalize



- ST\_M
- ST\_PerpPoints
- ST\_RemovePoint
- ST\_X
- ST\_Y
- ST\_Z

## ST\_AppendPoint

ST\_AppendPoint toma una curva y un punto como parámetros de entrada y extiende la curva por el punto determinado.

## ST\_ChangePoint

ST\_ChangePoint toma una curva y dos puntos como parámetros de entrada. Sustituye todas las apariciones del primer punto de la curva de entrada por el segundo punto y devuelve la curva resultante.

## ST\_Generalize

ST\_Generalize toma una geometría y un umbral como parámetros de entrada y representa dicha geometría con un número reducido de puntos, al mismo tiempo que conserva las características generales de la geometría. Se utiliza el algoritmo de simplificación de líneas de Douglas-Peucker, mediante el cual la secuencia de puntos que define la geometría se subdivide repetidamente hasta que un tramo de los puntos se pueda sustituir por un segmento lineal recto. En este segmento lineal, ninguno de los puntos que lo definen se aparta del segmento lineal recto más allá del umbral especificado. Las coordenadas Z y M no se tienen en cuenta para la simplificación.

## ST\_M

Si un punto determinado no tiene una medida asociada a él, ST\_M puede proporcionar una medida para guardarla con el punto. Si el punto tiene una medida asociada, ST\_M puede sustituir esa medida por otra.

## ST\_PerpPoints

ST\_PerpPoints toma como parámetros de entrada una curva o multicurva y un punto y devuelve la proyección perpendicular de ese punto en la curva o multicurva. Se obtiene el punto con la distancia más pequeña entre el punto proporcionado y el punto perpendicular. Si dos o más puntos proyectados perpendicularmente son equidistantes respecto al punto proporcionado, se devuelven todos ellos.

## ST\_RemovePoint

ST\_RemovePoint toma una curva y un punto como parámetros de entrada y devuelve dicha curva después de eliminar de ella todos los puntos que son iguales al punto especificado. Si la curva proporcionada tiene coordenadas Z o M, el punto también debe tener coordenadas Z o M.

## ST\_X

ST\_X puede sustituir la coordenada X de un punto por otra coordenada X.

## ST\_Y

ST\_Y puede sustituir la coordenada Y de un punto por otra coordenada Y.

## ST\_Z

Si un punto proporcionado carece de coordenada Z, ST\_Z puede añadir una coordenada Z al punto. Si el punto tiene una coordenada Z, ST\_Z puede sustituir esta coordenada por otra coordenada Z.

---

## Función que devuelve información sobre distancias

ST\_Distance toma como parámetros de entrada dos geometrías y, opcionalmente, una unidad de medida, y devuelve la distancia más corta entre cualquier punto de la primera geometría y cualquier punto de la segunda geometría, expresada en las unidades proporcionadas.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las dos geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

Por ejemplo, ST\_Distance podría notificar la distancia más corta que un avión debe recorrer entre dos localidades. La Figura 56 muestra esta información.

La figura muestra un mapa de Estados Unidos con una línea recta entre dos puntos denominados Los Angeles y Chicago.



Figura 56. Distancia mínima entre dos ciudades. ST\_Distance puede utilizar como entrada las coordenadas de las ciudades de Los Angeles y Chicago y obtener el valor correspondiente a la distancia mínima entre estas ciudades.

---

## Función que proporciona información sobre índices

ST\_GetIndexParams toma como parámetro de entrada el identificador de un índice espacial o columna espacial y devuelve los parámetros utilizados para definir el índice en la columna espacial. Opcionalmente se puede especificar un número de parámetro, en cuyo caso sólo se obtiene el parámetro identificado por el número.

---

## Conversiones entre sistemas de coordenadas

ST\_Transform toma como parámetros de entrada una geometría y un identificador de sistemas de referencia espacial y transforma la geometría para que esté representada en el sistema de referencia espacial especificado. Se realizan proyecciones y conversiones entre diferentes sistemas de coordenadas y se ajustan las coordenadas de las geometrías según convenga.

---

## Capítulo 23. Funciones espaciales: sintaxis y parámetros

Esta sección presenta las funciones espaciales descritas en las secciones siguientes. Trata acerca de algunos factores que son comunes para todas o para la mayoría de funciones espaciales. A continuación se documentan las funciones en orden alfabético.

---

### Funciones espaciales: consideraciones y tipos de datos asociados

Esta sección describe lo que necesita conocer para codificar funciones espaciales. Esta información comprende:

- Factores a tener en cuenta: la necesidad de especificar el esquema al que pertenecen las funciones espaciales y el hecho de que puede invocar a algunas funciones como métodos.
- Cómo abordar una situación en la que una función espacial no puede procesar el tipo de geometría devuelto por otra función espacial.
- Una tabla que muestra una lista de las funciones espaciales de acuerdo con el tipo de dato espacial utilizado como entrada

Cuando utilice funciones espaciales, tenga en cuenta estos factores:

- Para poder invocar a una función espacial, el nombre de la función debe estar calificado por el nombre del esquema al que pertenecen las funciones espaciales: DB2GSE. Una forma de hacer esto es especificar explícitamente el esquema en la sentencia de SQL donde se utiliza la función; por ejemplo:

```
SELECT db2gse.ST_Relate (g1, g2, 'T*F***FFF2') EQUALS FROM relate_test
```

Como método alternativo, para evitar especificar el esquema cada vez que se debe llamar a una función, puede añadir DB2GSE al registro especial CURRENT FUNCTION PATH. Para obtener los valores actuales de este registro especial, escriba este mandato de SQL:

```
VALUES CURRENT FUNCTION PATH
```

Para actualizar el registro especial CURRENT FUNCTION PATH con DB2GSE, emita este mandato de SQL:

```
set CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

- Puede invocar a algunas funciones espaciales como métodos. Por ejemplo, en el código siguiente, se invoca a ST\_Area primero como función y luego como método. En ambos casos, ST\_Area está codificado para actuar sobre un polígono cuyo ID es 10 y que está contenido en la columna SALES\_ZONE de la tabla STORES. Cuando se invoca, ST\_Area devolverá el área del elemento gráfico real que el polígono representa, Sales Zone número 10.

ST\_Area invocada como función:

```
SELECT ST_Area(sales_zone)
FROM stores
WHERE id = 10
```

ST\_Area invocada como método:

```
SELECT sales_zone..ST_Area()
FROM stores
WHERE id = 10
```

Las funciones `ST_BuildMBRAggr` y `ST_BuildUnionAggr` se describen en "MBR Aggregate" (Agregado MBR) y "Union Aggregate" (Agregado de unión), respectivamente.

## Factores a tener en cuenta

### Tratamiento de valores de `ST_Geometry` como valores de un subtipo

Si una función espacial devuelve una geometría cuyo tipo estático es un supertipo y la geometría se pasa a una función que sólo acepta geometrías de un tipo que esté subordinado a este supertipo, se emite una excepción de compilación.

Por ejemplo, el tipo estático del parámetro de salida de la función `ST_Union` es `ST_Geometry`, el supertipo de todos los tipos de datos espaciales. El parámetro de entrada estático de la función `ST_PointOnSurface` puede ser `ST_Polygon` o `ST_MultiPolygon`, dos subtipos de `ST_Geometry`. Si DB2® intenta pasar a `ST_PointOnSurface` las geometrías devueltas por `ST_Union`, DB2 emite la siguiente excepción en tiempo de compilación:

```
SQL00440N No se ha encontrado ninguna función
denominada "ST_POINTONSURFACE"
que tenga argumentos compatibles en la vía de función. SQLSTATE=42884
```

Este mensaje indica que DB2 no pudo encontrar una función llamada `ST_PointOnSurface` y que tenga un parámetro de entrada de tipo `ST_Geometry`.

Para permitir que las geometrías de un supertipo pasen a funciones que sólo aceptan subtipos del supertipo, utilice el operador `TREAT`. Tal como se indicó anteriormente, `ST_Union` devuelve geometrías cuyo tipo estático es `ST_Geometry`. También puede devolver geometrías cuyo subtipo dinámico es `ST_Geometry`. Por ejemplo, suponga que la función devuelve una geometría cuyo tipo dinámico es `ST_MultiPolygon`. En ese caso, el operador `TREAT` necesita que esta geometría se utilice con el tipo estático `ST_MultiPolygon`. Esto coincide con uno de los tipos de datos del parámetro de entrada de `ST_PointOnSurface`. Si `ST_Union` devuelve un valor `ST_MultiPolygon`, DB2 emite una excepción de ejecución.

Si una función devuelve una geometría de un supertipo, normalmente el operador `TREAT` puede indicar a DB2 que trate esa geometría como un subtipo de ese supertipo. Pero tenga en cuenta que esta operación solo es efectiva si el subtipo coincide con o está subordinado a un subtipo estático definido como parámetro de entrada de la función a la que se pasa la geometría. Si no se cumple esta condición, DB2 emite una excepción de ejecución.

Considere otro ejemplo: suponga que desea determinar los puntos perpendiculares para un punto determinado situado en el límite de un polígono que carece de huecos. Puede utilizar la función `ST_Boundary` para obtener el límite a partir del polígono. El parámetro de salida estático de `ST_Boundary` es `ST_Geometry`, pero `ST_PerpPoints` acepta geometrías `ST_Curves`. Debido a que todos los polígonos tienen como límite una cadena lineal (que es también una curva), y debido a que el tipo de datos Cadena Lineal (`ST_LineString`) está subordinado a `ST_Curve`, la operación siguiente permitirá que un polígono `ST_Geometry` devuelto por `ST_Boundary` se pase a `ST_PerpPoints`:

```
SELECT ST_AsText(ST_PerpPoints(TREAT(ST_Boundary(polygon) as ST_Curve)),
    ST_Point(30.5, 65.3, 1)))
FROM polygon_table
```

En lugar de invocar a ST\_Boundary y ST\_PerpPoints como funciones, puede invocarlos como métodos. Para ello, especifique este código:

```
SELECT TREAT(ST_Boundary(polygon) as ST_Curve)..
       ST_PerpPoints(St_Point(30.5, 65.3, ))..ST_AsText()
FROM   polygon_table
```

## Funciones espaciales de acuerdo con el tipo de datos de entrada

La Tabla 57 lista las funciones espaciales de acuerdo con el tipo de datos de entrada que pueden aceptar.

**Importante:** Tal como se indicó en otro lugar, los tipos de datos espaciales forman una jerarquía, cuya raíz es ST\_Geometry. Cuando la documentación de DB2 Spatial Extender indica que un valor de un supertipo de esta jerarquía se puede utilizar como entrada para una función, como alternativa también se puede utilizar como entrada para la función un valor de cualquier subtipo de ese supertipo.

Por ejemplo, las primeras de entradas de la Tabla 57 indican que ST\_Area y varias otras funciones pueden utilizar como entrada valores del tipo de datos ST\_Geometry. Por tanto, estas funciones también pueden utilizar como entrada valores de cualquier subtipo de ST\_Geometry: ST\_Point, ST\_Curve, ST\_LineString, etc.

Tabla 57. Funciones espaciales de acuerdo con el tipo de datos de entrada

Tipo de datos del parámetro de entrada	Función
ST_Geometry	EnvelopesIntersect
	ST_Area
	ST_AsBinary
	ST_AsGML
	ST_AsShape
	ST_AsText
	ST_Boundary
	ST_Buffer
	ST_BuildMBRAggr
	ST_BuildUnionAggr
	ST_Centroid
	ST_Contains
	ST_ConvexHull
	ST_CoordDim
	ST_Crosses
	ST_Difference
	ST_Dimension
	ST_Disjoint
	ST_Distance
	ST_Envelope
	ST_EnvIntersects
	ST_Equals
	ST_FindMeasure o ST_LocateAlong
	ST_Generalize
	ST_GeometryType

Tabla 57. Funciones espaciales de acuerdo con el tipo de datos de entrada (continuación)

Tipo de datos del parámetro de entrada	Función
ST_Geometry (continuación)	ST_Intersection ST_Intersects ST_Is3D ST_IsEmpty ST_IsMeasured ST_IsSimple ST_IsValid ST_MaxM ST_MaxX ST_MaxY ST_MaxZ ST_MBR ST_MBRIntersects ST_MeasureBetween o ST_LocateBetween ST_MinM ST_MinX ST_MinY ST_MinZ ST_NumPoints ST_Overlaps ST_Relate ST_SRID o ST_SrsId ST_SrsName ST_SymDifference ST_ToGeomColl ST_ToLineString ST_ToMultiLine ST_ToMultiPoint ST_ToMultiPolygon ST_ToPoint ST_ToPolygon ST_Touches ST_Transform ST_Union ST_Within
ST_Point	ST_M ST_X ST_Y ST_Z
ST_Curve	ST_AppendPoint ST_ChangePoint ST_EndPoint ST_IsClosed ST_IsRing ST_Length ST_MidPoint ST_PerpPoints ST_RemovePoint ST_StartPoint
ST_LineString	ST_PointN ST_Polygon

Tabla 57. Funciones espaciales de acuerdo con el tipo de datos de entrada (continuación)

Tipo de datos del parámetro de entrada	Función
ST_Surface	ST_Perimeter ST_PointOnSurface
ST_GeomCollection	ST_GeometryN ST_NumGeometries
ST_MultiPoint	ST_PointN
ST_MultiCurve	ST_IsClosed ST_Length ST_PerpPoints
ST_MultiLineString	ST_LineStringN ST_NumLineStrings ST_Polygon
ST_MultiSurface	ST_Perimeter ST_PointOnSurface
ST_MultiPolygon	ST_NumPolygons ST_PolygonN

## EnvelopesIntersect

EnvelopesIntersect acepta dos tipos de parámetros de entrada:

- Dos geometrías  
EnvelopesIntersect devuelve un 1 si la envoltura de la primera geometría forma una intersección con la envoltura de la segunda geometría. En caso contrario, se devuelve un 0 (cero).
- Una geometría, cuatro valores de coordenadas de tipo DOUBLE que definen los ángulos inferior izquierdo y superior derecho de una ventana rectangular y el identificador de sistemas de referencia espacial.  
EnvelopesIntersect devuelve un 1 si la envoltura de la primera geometría forma una intersección con la envoltura definida por los cuatro valores de tipo DOUBLE. En caso contrario, se devuelve un 0 (cero).

### Sintaxis

```
db2gse.EnvelopesIntersect(geometría1, geometría2 | rectangular-window)
```

#### rectangular-window:

```
| mín_x, mín_y, máx_x, máx_y, id_srs |
```

### Parámetros

*geometría1*

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría

cuya envoltura se debe verificar si forma intersección con la envoltura de *geometría2* o la ventana rectangular definida por los cuatro valores de tipo DOUBLE.

*geometría2*

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría cuya envoltura se debe verificar si forma intersección con la envoltura de *geometría1*.

*mín\_x*

Especifica el valor mínimo de la coordenada X para la envoltura. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

Para los datos geodésicos se aplican las siguientes normas:

- *mín\_x* debe ser un valor de longitud comprendido entre -180 Y 180 grados.
- *mín\_x* es mayor que *máx\_x* cuando la envoltura se solapa con el meridiano 180°.

*mín\_y*

Especifica el valor mínimo de la coordenada Y para la envoltura. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

Para los datos geodésicos se aplican las siguientes normas:

- *mín\_y* debe ser un valor de latitud comprendido entre -90 y 90 grados.
- *mín\_y* debe ser menor que el valor *máx\_y*.

*máx\_x*

Especifica el valor máximo de la coordenada X para la envoltura. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

Para los datos geodésicos se aplican las siguientes normas:

- *máx\_x* debe ser un valor de longitud comprendido entre -180 y 180 grados.
- *máx\_x* es menor que el valor *mín\_x* cuando la envoltura se solapa con el meridiano 180°.

*máx\_y*

Especifica el valor máximo de la coordenada Y para la envoltura. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

Para los datos geodésicos se aplican las siguientes normas:

- *máx\_y* debe ser un valor de latitud comprendido entre -90 y 90 grados.
- *máx\_y* debe ser mayor que el valor *mín\_y*.

*id\_srs*

Identifica de forma exclusiva el sistema de referencia espacial. El identificador de sistemas de referencia espacial debe coincidir con el identificador de sistemas de referencia espacial del parámetro de geometría. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es INTEGER.



## Tipo de retorno

INTEGER

## Ejemplo

En este ejemplo se crean dos polígonos que representan condados y se determina si alguno de ellos forma una intersección con un área geográfica especificada por los cuatro valores de tipo DOUBLE.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE condados (id INTEGER, nombre CHAR(20), geometría ST_Polygon)

INSERT INTO condados VALUES
(1, 'Condado_1', ST_Polygon('polígono((0 0, 30 0, 40 30, 40 35,
5 35, 5 10, 20 10, 20 5, 0 0))' ,0))

INSERT INTO condados VALUES
(2, 'Condado_2', ST_Polygon('polígono((15 15, 15 20, 60 20, 60 15,
15 15))' ,0))

INSERT INTO condados VALUES
(3, 'Condado_3', ST_Polygon('polígono((115 15, 115 20, 160 20, 160 15,
115 15))' ,0))

SELECT nombre
FROM condados as c
WHERE EnvelopesIntersect(c.geometry, 15, 15, 60, 20, 0) =1
```

Resultados:

```
Nombre
-----
Condado_1
Condado_2
```

---

## MBR Aggregate (Agregado MBR)

El uso combinado de las funciones ST\_BuildMBRAggr y ST\_GetAggrResult agrega una columna de geometrías de una columna seleccionada a una geometría individual creando un rectángulo que representa el rectángulo delimitador mínimo (minimum bounding rectangle, MBR) que encierra todas las geometrías contenidas en la columna. Las coordenadas Z y M se descartan cuando se calcula el agregado.

Si todas las geometrías que se deben combinar son nulas, se devuelve un valor nulo. Si todas las geometrías son nulas o están vacías, el resultado es una geometría vacía. Si el rectángulo delimitador mínimo de todas las geometrías a combinar da como resultado un punto, se devuelve este punto como valor de tipo ST\_Point. Si el rectángulo delimitador mínimo de todas las geometrías a combinar da como resultado una cadena lineal horizontal o vertical, se devuelve esta cadena lineal como valor de tipo ST\_LineString. En otro caso, el rectángulo delimitador mínimo se devuelve como valor de tipo ST\_Polygon.

## Sintaxis

```
►► db2gse.ST_GetAggrResult(—MAX—(——————►)
```

## Parámetro

### geometrías

Es una columna seleccionada que tiene un tipo de ST\_Geometry o uno de sus subtipos y representa todas las geometrías para las que se debe calcular el rectángulo delimitador mínimo.

## Tipo de retorno

db2gse.ST\_Geometry

## Restricciones

No puede crear el agregado de unión de una columna espacial en una selección completa en los casos siguientes:

- En un entorno de Database Partitioning Feature (DPF).
- Si se utiliza la cláusula GROUP BY en la selección completa.
- Si utiliza una función distinta de la función de agregación MAX de DB2.

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar la función ST\_BuildMBRAggr para obtener el rectángulo delimitador máximo de todas las geometrías contenidas en una columna. Este ejemplo añade varios puntos a la columna GEOMETRY de la tabla SAMPLE\_POINTS. Luego, el código de SQL determina el rectángulo delimitador máximo de todos los puntos puestos juntos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_points (id integer, geometry ST_Point)
```

```
INSERT INTO sample_points (id, geometry)
VALUES
```

```
(1, ST_Point(2, 3, 1)),
(2, ST_Point(4, 5, 1)),
(3, ST_Point(13, 15, 1)),
(4, ST_Point(12, 5, 1)),
(5, ST_Point(23, 2, 1)),
(6, ST_Point(11, 4, 1))
```

```
SELECT cast(ST_GetAggrResult(MAX(ST_BuildMBRAggr
    (geometry)))..ST_AsText AS varchar(160))
    AS ";Aggregate_of_Points";
FROM sample_points
```

Resultados:

```
Aggregate_of_Points
```

```
-----
POLYGON (( 2.00000000 2.00000000, 23.00000000 2.00000000,
23.00000000 15.00000000, 2.00000000 15.00000000, 2.00000000
2.00000000))
```

---

## ST\_AppendPoint

ST\_AppendPoint toma una curva y un punto como parámetros de entrada y extiende la curva por el punto determinado. Si la curva proporcionada tiene coordenadas Z o M, el punto también debe tener coordenadas Z o M. La curva resultante se representa según el sistema de referencia espacial de la curva proporcionada.

Si el punto a añadir no está representado según el mismo sistema de referencia espacial que la curva, se convertirá al otro sistema de referencia espacial.

Si la curva proporcionada es cerrada o simple, es posible que la curva resultante ya no sea cerrada ni simple. Si la curva o el punto proporcionado tienen un valor nulo, o si la curva está vacía, se devuelve un valor nulo. Si el punto que se debe añadir está vacío, se devuelve inalterada la curva de entrada y se emite un aviso (SQLSTATE 01HS3).

También se puede invocar esta función como método.

### Sintaxis

► db2gse.ST\_AppendPoint(—*curva*—, —*punto*—) ◀

### Parámetro

**curva** Valor de tipo ST\_Curve o uno de sus subtipos que representa la curva a la que se añade *punto*.

**punto** Valor de tipo ST\_Point que representa el punto que se añade a la *curva*.

### Tipo de retorno

db2gse.ST\_Curve

### Ejemplos

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este código crea dos cadenas lineales, cada una con tres puntos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines(id integer, line ST_Linestring)

INSERT INTO sample_lines VALUES
  (1, ST_LineString('linestring (10 10, 10 0, 0 0)', 0) )

INSERT INTO sample_lines VALUES
  (2, ST_LineString('linestring z (0 0 4, 5 5 5, 10 10 6)', 0) )
```

#### Ejemplo 1

Este ejemplo añade el punto (5, 5) al final de una cadena lineal.

```
SELECT CAST(ST_AsText(ST_AppendPoint(line, ST_Point(5, 5)))
  AS VARCHAR(120)) New
FROM sample_lines
WHERE id=1
```

Resultados:

NEW

```
-----  
LINESTRING ( 10.00000000 10.00000000, 10.00000000 0.00000000,  
0.00000000 0.00000000, 5.00000000 5.00000000)
```

## Ejemplo 2

Este ejemplo añade el punto (15, 15, 7) al final de una cadena lineal con coordenadas Z.

```
SELECT CAST(ST_AsText(ST_AppendPoint(line, ST_Point(15.0, 15.0, 7.0)))  
AS VARCHAR(160)) New  
FROM sample_lines  
WHERE id=2
```

Resultados:

NEW

```
-----  
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 5.00000000  
5.00000000 5.00000000, 10.00000000 10.00000000 6.00000000,  
15.00000000 15.00000000 7.00000000)
```

---

## ST\_Area

ST\_Area toma como parámetros de entrada una geometría y, opcionalmente, una unidad de medida, y devuelve el área cubierta por la geometría, expresada en la unidad de medida proporcionada.

Si la geometría es un polígono o un multipolígono, se devuelve el área cubierta por la geometría. El área correspondiente a puntos, cadenas lineales, multipuntos y multilíneas es 0 (cero). Si la geometría es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
db2gse.ST_Area(—geometría [,—unidad]—)
```

### Parámetros

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que define el área.

#### unidad

Valor de tipo VARCHAR(128) que identifica las unidades utilizadas para medir el área. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST\_UNITS\_OF\_MEASURE.

Si se omite el parámetro *unidad*, se siguen las normas siguientes para determinar la unidad utilizada para medir el área:

- Si *geometría* está en un sistema de coordenadas proyectadas o geocéntricas, se utiliza la unidad lineal correspondiente a este sistema de coordenadas.

- Si *geometría* está en un sistema de coordenadas geográficas pero no está en un sistema de referencia espacial (SRS) geodésico, se utiliza la unidad angular asociada a este sistema de coordenadas.
- Si la *geometría* está en un SRS geodésico, la unidad de medida por omisión será el metro cuadrado.

**Restricciones para las conversiones de unidades:** Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La geometría está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La geometría está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La geometría está en un sistema de coordenadas geográficas, pero no está en un SRS geodésico y se especifica una unidad lineal.
- La geometría está en un sistema de coordenadas geográficas, está en un SRS geodésico y se especifica una unidad lineal.

## Tipo de retorno

DOUBLE

## Ejemplos

### Ejemplo 1

Necesitamos conocer el área cubierta por cada región de ventas de la empresa. Los polígonos correspondientes a las regiones de ventas están contenidos en la tabla SAMPLE\_POLYGONS. El área se calcula aplicando la función ST\_Area a la columna de geometrías.

```
db2se create_srs se_bank -srsId 4000 -srsName new_york1983 -xOffset 0
      -yOffset 0 -xScale 1 -yScale 1
      -coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet
```

```
CREATE TABLE sample_polygons (id INTEGER, geometry ST_POLYGON)
```

```
INSERT INTO sample_polygons (id, geometry)
VALUES
  (1, ST_Polygon('polygon((0 0, 0 10, 10 10, 10 0, 0 0))', 4000) ),
  (2, ST_Polygon('polygon((20 0, 30 20, 40 0, 20 0))', 4000) ),
  (3, ST_Polygon('polygon((20 30, 25 35, 30 30, 20 30))', 4000))
```

La sentencia SELECT siguiente recupera el ID y el área de la región de ventas:

```
SELECT id, ST_Area(geometry) AS area
FROM   sample_polygons
```

Resultados:

ID	AREA
1	+1.00000000000000E+002
2	+2.00000000000000E+002
3	+2.50000000000000E+001

### Ejemplo 2

La sentencia SELECT siguiente recupera el ID y el área de la región de ventas expresada en diversas unidades:

```

SELECT id,
       ST_Area(geometry) square_feet,
       ST_Area(geometry, 'METER') square_meters,
       ST_Area(geometry, 'STATUTE MILE') square_miles
FROM   sample_polygons

```

Resultados:

ID	SQUARE_FEET	SQUARE_METERS	SQUARE_MILES
1	+1.000000000000000E+002	+9.29034116132748E+000	+3.58702077598427E-006
2	+2.000000000000000E+002	+1.85806823226550E+001	+7.17404155196855E-006
3	+2.500000000000000E+001	+2.32258529033187E+000	+8.96755193996069E-007

### Ejemplo 3

Este ejemplo determina el área de un polígono que está definido mediante coordenadas de Plano de Estado.

Con el siguiente mandato se crea el sistema de referencia espacial de Plano de estado con un ID de 3:

```

db2se create_srs SAMP_DB -srsId 3 -srsName z3101a -xoffset 0
      -yOffset 0 -xScale 1 -yScale 1
      -coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet

```

Las siguientes sentencias de SQL añaden el polígono del sistema de referencia espacial 3 a la tabla y determinan el área en pies cuadrados, metros cuadrados y millas cuadradas.

```

SET current function path db2gse;
CREATE TABLE Sample_Poly3 (id integer, geometry ST_Polygon);
INSERT into Sample_Poly3 VALUES
  (1, ST_Polygon('polygon((567176.0 1166411.0,
                        567176.0 1177640.0,
                        637948.0 1177640.0,
                        637948.0 1166411.0,
                        567176.0 1166411.0 ))', 3));
SELECT id, ST_Area(geometry) "Square Feet",
       ST_Area(geometry, 'METER') "Square Meters",
       ST_Area(geometry, 'STATUTE MILE') "Square Miles"
FROM   Sample_Poly3;

```

Resultados:

ID	Pies cuadrados	Metros cuadrados	Millas cuadradas
1	+7.94698788000000E+008	+7.38302286101346E+007	+2.85060106320552E+001

### Ejemplo 4

El analista espacial necesita una lista del área cubierta por cada región de exploración. Los polígonos de región de exploración se almacenan en la tabla SAMPLE\_GEODETIC\_TAB e incluyen las siguientes regiones:

- Una región alrededor del polo norte
- Una región alrededor del polo sur
- Una región que ocupa el meridiano 180°

El segundo campo del siguiente archivo de entrada, samp\_wkt\_rows.txt, contiene polígonos que representan estas regiones:

```

1|'polygon((5 82,15 82,25 82,35 82,45 82,55 82,65 82,75 82,85 82,95 82,
105 82,115 82,125 82,135 82,145 82,155 82,165 82,175 82,-175 82,-165 82,
-155 82,-145 82,-135 82,-125 82,-115 82,-105 82,-95 82,-85 82,-75 82,

```

```

-65 82,-55 82,-45 82,-35 82,-25 82,-15 82,-5 82,5 82))'|'North Pole region'
2|'polygon((175 -82,165 -82,155 -82,145 -82,135 -82,125 -82,115 -82,
105 -82,95 -82,85 -82,75 -82,65 -82,55 -82,45 -82,35 -82,25 -82,15 -82,
5 -82,-5 -82,-15 -82,-25 -82,-35 -82,-45 -82,-55 -82,-65 -82,-75 -82,
-85 -82,-95 -82,-105 -82,-115 -82,-125 -82,-135 -82,-145 -82,-155 -82,
-165 -82,-175 -82,175 -82))'|'South Pole region'
3|'polygon((-175 -42,-175 1,-175 42,175 42,175 -1,175 -42,-175 -42))
'|'180th meridian'

```

Las siguientes sentencias de SQL añaden los polígonos del sistema de referencia espacial geodésico 2000000000 a la tabla SAMPLE\_GEODETIC\_TAB.

```

SET current function path db2gse;
CREATE TABLE db2se_samp.gsege_temp_samp (
    gid        INTEGER,
    g1_wkt     varchar(500),
    comment    varchar(255)
) NOT LOGGED INITIALLY;
LOAD FROM samp_wkt_rows.txt OF DEL MODIFIED BY CHARDEL'' COLDEL|
INSERT INTO db2se_samp.gsege_temp_samp;

CREATE TABLE sample_geodetic_tab
(gid INTEGER NOT NULL PRIMARY KEY,
 geometry ST_Geometry),
comment varchar(255));

INSERT INTO sample_geodetic_tab
SELECT gid, ST_GeomFromText(g1_wkt, 2000000000), comment
FROM db2se_samp.gsege_temp_samp;

```

La función ST\_Area calcula el área del polígono en la columna de geometría. La unidad de medida por omisión de ST\_Area es el metro cuadrado. La siguiente sentencia SELECT recupera el ID y el área de la región de exploración expresada en metros cuadrados, pies cuadrados y millas cuadradas.

```

SELECT id, ST_Area(geometry) AS SQUARE_METERS,
ST_Area(geometry,'FOOT') AS SQUARE_FEET,
ST_Area(geometry, 'STATUTE MILE') AS SQUARE_MILES
FROM sample_geodetic_tab
WHERE id BETWEEN 1 AND 9 ORDER BY id;

```

ID	SQUARE_METERS	SQUARE_FEET	SQUARE_MILES
1	+2.52472719957839E+012	+2.71759374028922E+013	+9.74802621488040E+005
2	+2.52475431563494E+012	+2.71762292776957E+013	+9.74813091056005E+005
3	+9.43568029137069E+012	+1.01564817377028E+014	+3.64313652781464E+006

## ST\_AsBinary

ST\_AsBinary toma una geometría como parámetro de entrada y devuelve su presentación binaria convencional. Las coordenadas Z y M se descartan y no estarán representadas en la representación binaria convencional.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_AsBinary—(—geometría—)—————◄◄

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que se convertirá a la representación binaria convencional.

## Tipo de retorno

BLOB(2G)

## Ejemplos

El código siguiente muestra cómo utilizar la función ST\_AsBinary para convertir los puntos contenidos en las columnas de geometrías de la tabla SAMPLE\_POINTS en una representación binaria convencional (WKB) en la columna BLOB.

```
CREATE TABLE SAMPLE_POINTS (id integer, geometry ST_POINT, wkb BLOB(32K))
```

```
INSERT INTO SAMPLE_POINTS (id, geometry)
VALUES
    (1100, ST_Point(10, 20, 1))
```

### Ejemplo 1

Este ejemplo llena la columna binaria (WKB), cuyo ID es 1111, a partir de la columna GEOMETRY, cuyo ID es 1100.

```
INSERT INTO sample_points(id, wkb)
VALUES (1111,
    (SELECT ST_AsBinary(geometry)
     FROM sample_points
     WHERE id = 1100))
```

```
SELECT id, cast(ST_Point(wkb)..ST_AsText AS varchar(35)) AS point
FROM sample_points
WHERE id = 1111
```

Resultados:

ID	Point
1111	POINT ( 10.00000000 20.00000000)

### Ejemplo 2

Este ejemplo muestra la representación binaria WKB.

```
SELECT id, substr(ST_AsBinary(geometry), 1, 21) AS point_wkb
FROM sample_points
WHERE id = 1100
```

Resultados:

ID	POINT_WKB
1100	x'0101000000000000000000000024400000000000003440'



---

## ST\_AsGML

ST\_AsGML toma una geometría como parámetro de entrada y devuelve su representación utilizando el lenguaje GML (Geography Markup Language).

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
db2gse.ST_AsGML(geometría, gmlLevelentero)
```

### Parámetro

#### gmlLevel

Es un parámetro opcional que especifica el nivel de especificación GML que se usará para dar formato a los datos GML que se devolverán. Los valores válidos son:

- 2 - Utilice el nivel de especificación GML 2 con la etiqueta <gml:coordinates>.
- 3 - Utilice el nivel de especificación GML 3 con la etiqueta <gml:poslist>.

Si no se especifica ningún parámetro, la salida se devuelve utilizando el nivel de especificación GML 2 con la etiqueta <gml:coord>.

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que se convertirá en la correspondiente representación GML.

### Tipo de retorno

CLOB(2G)

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El fragmento de código siguiente muestra cómo utilizar la función ST\_AsGML para visualizar el fragmento GML. Este ejemplo llena la columna GML, a partir de la columna de geometrías cuyo ID es 2222.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE SAMPLE_POINTS (id integer, geometry ST_POINT, gml CLOB(32K))

INSERT INTO SAMPLE_POINTS (id, geometry)
VALUES
    (1100, ST_Point(10, 20, 1))

INSERT INTO sample_points(id, gml)
VALUES (2222,
    (SELECT ST_AsGML(geometry)
     FROM sample_points
     WHERE id = 1100))
```

La sentencia SELECT siguiente lista el ID y las representaciones GML de la geometría.

```
SELECT id, cast(ST_AsGML(geometry) AS varchar(110)) AS gml_fragment
FROM sample_points
WHERE id = 1100
```

Resultados:

```
SELECT id,
       cast(ST_AsGML(geometry) AS varchar(110)) AS gml,
       cast(ST_AsGML(geometry,2) AS varchar(110)) AS gml2,
       cast(ST_AsGML(geometry,3) AS varchar(110)) AS gml3
FROM sample_points
WHERE id = 1100
```

La sentencia SELECT devuelve el resultado siguiente:

ID	GML	GML2	GML3
1100	<gml:Point srsName="EPSG:4269">  <gml:coord> <gml:X>10</gml:X><gml:Y>20</gml:Y> </gml:coord></gml:Point>	<gml:Point srsName="EPSG:4269">  <gml:coordinates> 10,20 </gml:coordinates></gml:Point>	<gml:Point srsName="EPSG:4269 srsDimension="2"> <gml:pos> 10,20 </gml:pos></gml:Point>

## ST\_AsShape

St\_AsShape toma una geometría como parámetro de entrada y devuelve su presentación de forma ESRI.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
►► db2gse.ST_AsShape(—geometría—)◄◄
```

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que se convertirá a la representación de forma ESRI correspondiente.

### Tipo de retorno

BLOB(2G)

### Ejemplo

El código siguiente muestra cómo utilizar la función ST\_AsShape para convertir los puntos contenidos en la columna de geometrías de la tabla SAMPLE\_POINTS en una representación binaria de formas en la columna BLOB de formas. Este ejemplo llena la columna de formas a partir de la columna de geometrías. La representación binaria de formas se utiliza para visualizar las geometrías en geonavegadores, los cuales necesitan que las geometrías se ajusten al formato del archivo de formas ESRI, o para crear las geometrías para el archivo \*.SHP del archivo de formas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE SAMPLE_POINTS (id integer, geometry ST_POINT, shape BLOB(32K))
```

```

INSERT INTO SAMPLE_POINTS (id, geometry)
VALUES
    (1100, ST_Point(10, 20, 1))

INSERT INTO sample_points(id, shape)
VALUES (2222,
    (SELECT ST_AsShape(geometry)
    FROM sample_points
    WHERE id = 1100))

SELECT id, substr(ST_AsShape(geometry), 1, 20) AS shape
FROM sample_points
WHERE id = 1100

```

Resultado:

```

ID      SHAPE
-----
1100   x'01000000000000000000000024400000000000003440'

```

---

## ST\_AsText

ST\_AsText toma una geometría como parámetro de entrada y devuelve su representación de texto convencional.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```

▶▶—db2gse.ST_AsText—(—geometría—)—▶▶

```

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que se convertirá a la representación de texto convencional correspondiente.

### Tipo de retorno

CLOB(2G)

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Después de capturar e insertar los datos en la tabla SAMPLE\_GEOMETRIES, el analista verifica la corrección de los valores insertados examinando la representación de texto convencional de las geometrías.

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

```

```

CREATE TABLE sample_geometries(id SMALLINT, spatial_type varchar(18),
    geometry ST_GEOMETRY)

```

```

INSERT INTO sample_geometries(id, spatial_type, geometry)

```

```
VALUES
(1, 'st_point', ST_Point(50, 50, 0)),
(2, 'st_linestring', ST_LineString('linestring
(200 100, 210 130, 220 140)', 0)),
(3, 'st_polygon', ST_Polygon('polygon((110 120, 110 140,
130 140, 130 120, 110 120))', 0))
```

La sentencia SELECT siguiente lista el tipo espacial y la representación de texto convencional (WKT) de las geometrías. La geometría se convierte a texto mediante la función ST\_AsText. Luego se convierte al tipo varchar(120) debido a que los datos de salida de la función ST\_AsText son de tipo CLOB(2G).

```
SELECT id, spatial_type, cast(geometry..ST_AsText
AS varchar(150)) AS wkt
FROM sample_geometries
```

Resultados:

ID	SPATIAL_TYPE	WKT
1	st_point	POINT ( 50.00000000 50.00000000)
2	st_linestring	LINestring ( 200.00000000 100.00000000, 210.00000000 130.00000000, 220.00000000 140.00000000)
3	st_polygon	POLYGON (( 110.00000000 120.00000000, 130.00000000 120.00000000, 130.00000000 140.00000000, 110.00000000140.00000000, 110.00000000 120.00000000))

## ST\_Boundary

ST\_Boundary toma una geometría como parámetro de entrada y devuelve su perímetro en forma de nueva geometría. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si la geometría proporcionada es un punto, un multipunto, una curva cerrada o una multicurva cerrada, o si está vacía, el resultado es una geometría vacía de tipo ST\_Point. Para las curvas o multicurvas que no son cerradas, los puntos de inicio y final de las curvas se proporcionan en forma de valor de tipo ST\_MultiPoint, a menos que dicho punto sea el punto inicial o final de un número par de curvas. Para las superficies y multisuperficies, se obtiene la curva que define el perímetro de la geometría proporcionada, en forma de valor de tipo ST\_Curve o ST\_MultiCurve. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

Si es posible, el tipo específico de la geometría devuelta será ST\_Point, ST\_LineString o ST\_Polygon. Por ejemplo, el perímetro de un polígono sin huecos es una cadena lineal individual, representada como ST\_LineString. El perímetro de un polígono con uno o varios huecos consiste en varias cadenas lineales, representadas como ST\_MultiLineString.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_Boundary—(—geometría—)—————►►

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos. Se devuelve el perímetro de esta geometría.

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea varias geometrías y determina el perímetro de cada geometría.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
  (1, ST_Polygon('polygon((40 120, 90 120, 90 150, 40 150, 40 120))', 0))

INSERT INTO sample_geoms VALUES
  (2, ST_Polygon('polygon((40 120, 90 120, 90 150, 40 150, 40 120),
    (70 130, 80 130, 80 140, 70 140, 70 130))', ,0))

INSERT INTO sample_geoms VALUES
  (3, ST_Geometry('linestring(60 60, 65 60, 65 70, 70 70)', ,0))

INSERT INTO sample_geoms VALUES
  (4, ST_Geometry('multilinestring((60 60, 65 60, 65 70, 70 70),
    (80 80, 85 80, 85 90, 90 90),
    (50 50, 55 50, 55 60, 60 60))', ,0))

INSERT INTO sample_geoms VALUES
  (5, ST_Geometry('point(30 30)', ,0))

SELECT id, CAST(ST_AsText(ST_Boundary(geometry)) as VARCHAR(320)) Boundary
FROM sample_geoms
```

### Resultados

```
ID      BOUNDARY
-----
1  LINESTRING ( 40.00000000 120.00000000, 90.00000000 120.00000000,
    90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
    120.00000000)

2  MULTILINESTRING (( 40.00000000 120.00000000, 90.00000000 120.00000000,
    90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
    120.00000000),( 70.00000000 130.00000000, 70.00000000 140.00000000,
    80.00000000 140.00000000, 80.00000000 130.00000000, 70.00000000
    130.00000000))

3  MULTIPOINT ( 60.00000000 60.00000000, 70.00000000 70.00000000)

4  MULTIPOINT ( 50.00000000 50.00000000, 70.00000000 70.00000000,
    80.00000000 80.00000000, 90.00000000 90.00000000)

5  POINT EMPTY
```

---

## ST\_Buffer

ST\_Buffer toma como parámetros de entrada una geometría, una distancia y, opcionalmente, una unidad de medida y devuelve una geometría que engloba a la geometría proporcionada y está separada de ella por la distancia especificada, expresada en la unidad indicada.

Cada punto del perímetro de la geometría resultante está separado de la geometría proporcionada por la distancia especificada. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Para los datos geodésicos, si especifica una distancia negativa, ST\_Buffer devuelve una región que va más allá de la distancia especificada de todos los puntos de la geometría de entrada. En otras palabras, una distancia negativa devuelve la región complementaria.

Los tramos curvos del perímetro de la geometría resultante se representan mediante cadenas lineales. Por ejemplo, la zona de separación alrededor de un punto, que daría lugar a una región circular, se representa mediante un polígono cuyo perímetro es una cadena lineal.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
db2gse.ST_Buffer(—geometría—, —distancia— [—unidad—])
```

### Parámetro

#### **geometría**

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la cual se debe crear la zona de separación. Para los datos geodésicos, ST\_Buffer sólo da soporte a los tipos de datos ST\_Point y ST\_MultiPoint.

#### **distancia**

Valor de tipo DOUBLE PRECISION que especifica la distancia que se debe utilizar para la zona de separación que debe haber alrededor de *geometría*. Para los datos geodésicos, la distancia no debe ser mayor que la del radio ecuatorial de la tierra. Para el elipsoide WGS-84, esta longitud es de 6378137,0 metros.

#### **unidad**

Valor de tipo VARCHAR(128) que identifica la unidad utilizada para medir la *distancia*. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST\_UNITS\_OF\_MEASURE.

Si se omite el parámetro *unidad*, se utilizan las siguientes normas para determinar la unidad de medida que se utiliza para la distancia:

- Si *geometría* está en un sistema de coordenadas proyectadas o geocéntricas, la unidad lineal asociada a este sistema de coordenadas es el valor por omisión.

- Si *geometría* está en un sistema de coordenadas geográficas, pero no está en un sistema de referencia espacial (SRS) geodésico, la unidad angular asociada a este sistema de coordenadas es el valor por omisión.
- Si *geometría* está en un SRS geodésico, la unidad de medida por omisión es el metro.

**Restricciones para las conversiones de unidades:** Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La geometría está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La geometría está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La geometría está en un sistema de coordenadas geográficas, pero no está en un SRS geodésico y se especifica una unidad lineal.
- La geometría está en un sistema de coordenadas geográficas, está en un SRS geodésico y se especifica una unidad lineal.

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplos

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

### Ejemplo 1

El código siguiente crea un sistema de referencia espacial, y crea y llena la tabla SAMPLE\_GEOMETRIES.

```
db2se create_srs se_bank -srsId 4000 -srsName new_york1983
      -xOffset 0 -yOffset 0 -xScale 1 -yScale 1
      -coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE
  sample_geometries (id INTEGER, spatial_type varchar(18),
  geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, spatial_type, geometry)
VALUES
  (1, 'st_point', ST_Point(50, 50, 4000)),
  (2, 'st_linestring',
  ST_LineString('linestring(200 100, 210 130,
  220 140)', 4000)),
  (3, 'st_polygon',
  ST_Polygon('polygon((110 120, 110 140, 130 140,
  130 120, 110 120))',4000)),
  (4, 'st_multipolygon',
  ST_MultiPolygon('multipolygon(((30 30, 30 40,
  35 40, 35 30, 30 30),(35 30, 35 40, 45 40,
  45 30, 35 30)))', 4000))
```

### Ejemplo 2

La sentencia SELECT siguiente utiliza la función ST\_Buffer para aplicar una zona de separación cuyo valor es 10.

```
SELECT id, spatial_type,
       cast(geometry..ST_Buffer(10)..ST_AsText AS varchar(470)) AS buffer_10
FROM sample_geometries
```

Resultados:

ID	SPATIAL_TYPE	BUFFER_10
1	st_point	POLYGON (( 60.00000000 50.00000000, 59.00000000 55.00000000, 54.00000000 59.00000000, 49.00000000 60.00000000, 44.00000000 58.00000000, 41.00000000 53.00000000, 40.00000000 48.00000000, 42.00000000 43.00000000, 47.00000000 41.00000000, 52.00000000 40.00000000, 57.00000000 42.00000000, 60.00000000 50.00000000))
2	st_linestring	POLYGON (( 230.00000000 140.00000000, 229.00000000 145.00000000, 224.00000000 149.00000000, 219.00000000 150.00000000, 213.00000000 147.00000000, 203.00000000 137.00000000, 201.00000000 133.00000000, 191.00000000 103.00000000, 191.00000000 99.00000000, 192.00000000 95.00000000, 196.00000000 91.00000000, 200.00000000 91.00000000, 204.00000000 91.00000000, 209.00000000 97.00000000, 218.00000000 124.00000000, 227.00000000 133.00000000, 230.00000000 140.00000000))
3	st_polygon	POLYGON (( 140.00000000 120.00000000, 140.00000000 140.00000000, 139.00000000 145.00000000, 130.00000000 150.00000000, 110.00000000 150.00000000, 105.00000000 149.00000000, 100.00000000 140.00000000, 100.00000000 120.00000000, 101.00000000 115.00000000, 110.00000000 110.00000000, 130.00000000 110.00000000, 135.00000000 111.00000000, 140.00000000 120.00000000))
4	st_multipolygon	POLYGON (( 55.00000000 30.00000000, 55.00000000 40.00000000, 54.00000000 45.00000000, 45.00000000 50.00000000, 30.00000000 50.00000000, 25.00000000 49.00000000, 20.00000000 40.00000000, 20.00000000 30.00000000, 21.00000000 25.00000000, 30.00000000 20.00000000, 45.00000000 20.00000000, 55.00000000 21.00000000, 55.00000000 30.00000000))

### Ejemplo 3

La sentencia SELECT siguiente utiliza la función ST\_Buffer para aplicar una zona de separación negativa cuyo valor es 5.

```
SELECT id, spatial_type,
       cast(ST_AsText(ST_Buffer(geometry, -5)) AS varchar(150))
       AS buffer_negative_5
FROM sample_geometries
WHERE id = 3
```

Resultados:

ID	SPATIAL_TYPE	BUFFER_NEGATIVE_5
3	st_polygon	POLYGON (( 115.00000000 125.00000000, 125.00000000 125.00000000, 125.00000000 135.00000000, 115.00000000 135.00000000, 115.00000000 125.00000000))

### Ejemplo 4

La sentencia SELECT siguiente muestra el resultado de aplicar un almacenamiento intermedio con el parámetro de unidad especificado.



```

SELECT id, spatial_type,
       cast(ST_AsText(ST_Buffer(geometry, 10, 'METER')) AS varchar(680))
       AS buffer_10_meter
FROM sample_geometries
WHERE id = 3

```

Resultados:

ID	SPATIAL_TYPE	BUFFER_10_METER
3	st_polygon	POLYGON (( 163.00000000 120.00000000, 163.00000000 140.00000000, 162.00000000 149.00000000, 159.00000000 157.00000000, 152.00000000 165.00000000, 143.00000000 170.00000000, 130.00000000 173.00000000, 110.00000000 173.00000000, 101.00000000 172.00000000, 92.00000000 167.00000000, 84.00000000 160.00000000, 79.00000000 151.00000000, 77.00000000 140.00000000, 77.00000000 120.00000000, 78.00000000 111.00000000, 83.00000000 102.00000000, 90.00000000 94.00000000, 99.00000000 89.00000000, 110.00000000 87.00000000, 130.00000000 87.00000000, 139.00000000 88.00000000, 147.00000000 91.00000000, 155.00000000 98.00000000, 160.00000000 107.00000000, 163.00000000 120.00000000))

## ST\_Centroid

ST\_Centroid toma una geometría como parámetro de entrada y devuelve el centro geográfico (el centro del rectángulo delimitador mínimo de la geometría) en forma de punto. El punto resultante se representa utilizando el sistema de referencia espacial de la geometría proporcionada.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_Centroid—(—geometría—)◀◀

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la cual se debe determinar el centro geométrico.

### Tipo de retorno

db2gse.ST\_Point

### Ejemplo

Este ejemplo crea dos geometrías y encuentra el centroide de las mismas.

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

```

```

INSERT INTO sample_geoms VALUES
(1, ST_Polygon('polygon
((40 120, 90 120, 90 150, 40 150, 40 120),
(50 130, 80 130, 80 140, 50 140, 50 130))',0))

```

```

INSERT INTO sample_geoms VALUES

```

```
(2, ST_MultiPoint('multipoint(10 10, 50 10, 10 30)' ,0))
```

```
SELECT id, CAST(ST_AsText(ST_Centroid(geometry))
as VARCHAR(40)) Centroid
FROM sample_geoms
```

Resultados:

ID	CENTROID
1	POINT ( 65.00000000 135.00000000)
2	POINT ( 30.00000000 20.00000000)

---

## ST\_ChangePoint

ST\_ChangePoint toma una curva y dos puntos como parámetros de entrada. Sustituye todas las apariciones del primer punto de la curva de entrada por el segundo punto y devuelve la curva resultante. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si los dos puntos no están representados en el mismo sistema de referencia espacial que la curva, se convertirán al sistema de referencia espacial utilizado para la curva.

Si la curva de entrada está vacía, el resultado es un valor vacío. Si la curva de entrada es un valor nulo, o cualquiera de los puntos de entrada es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
db2gse.ST_ChangePoint(—curve—, —punto_antiguo—, —punto_nuevo—)
```

### Parámetro

**curva** Valor de tipo ST\_Curve o un de sus subtipos que representa la curva en la que los puntos identificados como *punto\_antiguo* se cambian a *punto\_nuevo*.

**punto\_antiguo**

Valor de tipo ST\_Point que identifica los puntos de la curva que se cambian a *punto\_nuevo*.

**punto\_nuevo**

Valor de tipo ST\_Point que representa las nuevas ubicaciones de los puntos de la curva identificados por *punto\_antiguo*.

### Tipo de retorno

db2gse.ST\_Curve

### Restricciones

El punto que se debe cambiar en la curva debe ser uno de los puntos utilizados para definir la curva.

Si la curva tiene coordenadas Z o M, dichos puntos también deben tener coordenadas Z o M.

## Ejemplos

### Ejemplo 1

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente crea y llena la tabla SAMPLE\_LINES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines(id INTEGER, line ST_LineString)

INSERT INTO sample_lines VALUES
  (1, ST_LineString('linestring (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0) )

INSERT INTO sample_lines VALUES
  (2, ST_LineString('linestring z (0 0 4, 5 5 5, 10 10 6, 5 5 7)', 0) )
```

### Ejemplo 2

Este ejemplo cambia todas las apariciones del punto (5, 5) por el punto (6, 6) en la cadena lineal.

```
SELECT cast(ST_AsText(ST_ChangePoint(line, ST_Point(5, 5),
                                     ST_Point(6, 6))) as VARCHAR(160))
FROM sample_lines
WHERE id=1
```

### Ejemplo 3

Este ejemplo cambia todas las apariciones del punto (5, 5, 5) por el punto (6, 6, 6) en la cadena lineal.

```
SELECT cast(ST_AsText(ST_ChangePoint(line, ST_Point(5.0, 5.0, 5.0),
                                     ST_Point(6.0, 6.0, 6.0) )) as VARCHAR(180))
FROM sample_lines
WHERE id=2
```

Resultados:

```
NEW
-----
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 6.00000000 6.00000000
6.00000000, 10.00000000 10.00000000 6.00000000, 5.00000000 5.00000000
7.00000000)
```

---

## ST\_Contains

ST\_Contains toma dos geometrías como parámetros de entrada y devuelve un 1 si la primera geometría contiene completamente a la segunda; de lo contrario devuelve un 0 (cero) para indicar que la primera geometría no contiene completamente a la segunda.

Si cualquiera de las geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

## Sintaxis

► db2gse.ST\_Contains(—geometría1—,—geometría2—) ◀

## Parámetro

### geometría1

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la que se comprobar si contiene completamente a *geometría2*.

### geometría2

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la que se comprobar si está completamente dentro de *geometría1*.

**Restricciones:** Para los datos geodésicos, ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

## Tipo de retorno

INTEGER

## Ejemplos

### Ejemplo 1

El código siguiente crea y llena estas tablas:

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_points(id SMALLINT, geometry ST_POINT)

CREATE TABLE sample_lines(id SMALLINT, geometry ST_LINESTRING)

CREATE TABLE sample_polygons(id SMALLINT, geometry ST_POLYGON)

INSERT INTO sample_points (id, geometry)
VALUES
  (1, ST_Point(10, 20, 1)),
  (2, ST_Point('point(41 41)', 1))

INSERT INTO sample_lines (id, geometry)
VALUES
  (10, ST_LineString('linestring (1 10, 3 12, 10 10)', 1) ),
  (20, ST_LineString('linestring (50 10, 50 12, 45 10)', 1) )
INSERT INTO sample_polygons(id, geometry)
VALUES
  (100, ST_Polygon('polygon((0 0, 0 40, 40 40, 40 0, 0 0))', 1) )
```

### Ejemplo 2

El fragmento de código siguiente utiliza la función ST\_Contains para determinar qué puntos están incluidos en un polígono determinado.

```
SELECT poly.id AS polygon_id,
       CASE ST_Contains(poly.geometry, pts.geometry)
         WHEN 0 THEN 'does not contain'
         WHEN 1 THEN 'does contain'
       END AS contains,
       pts.id AS point_id
FROM   sample_points pts, sample_polygons poly
```

Resultados:

POLYGON_ID CONTAINS	POINT_ID
100 does contain	1
100 does not contain	2

### Ejemplo 3

El fragmento de código siguiente utiliza la función ST\_Contains para determinar qué líneas están incluidas en un polígono determinado.

```
SELECT poly.id AS polygon_id,  
       CASE ST_Contains(poly.geometry, line.geometry)  
         WHEN 0 THEN 'does not contain'  
         WHEN 1 THEN 'does contain'  
       END AS contains,  
       line.id AS line_id  
FROM   sample_lines line, sample_polygons poly
```

Resultados:

POLYGON_ID CONTAINS	LINE_ID
100 does contain	10
100 does not contain	20

---

## ST\_ConvexHull

ST\_ConvexHull utiliza una geometría como parámetro de entrada y devuelve su envoltura convexa.

La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si es posible, el tipo específico de la geometría devuelta será ST\_Point, ST\_LineString o ST\_Polygon. Por ejemplo, el perímetro de un polígono sin huecos es una cadena lineal individual, representada como ST\_LineString. El perímetro de un polígono con uno o varios huecos consiste en varias cadenas lineales, representadas como ST\_MultiLineString.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_ConvexHull—(*—geometría—*)—◄◄

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la cual se debe calcular la envoltura convexa.

### Tipo de retorno

db2gse.ST\_Geometry

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente crea y llena la tabla SAMPLE\_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id INTEGER, spatial_type varchar(18),
    geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, spatial_type, geometry)
VALUES
(1, 'ST_LineString', ST_LineString
('linestring(20 20, 30 30, 20 40, 30 50)', 0)),
(2, 'ST_Polygon', ST_Polygon('polygon
((110 120, 110 140, 120 130, 110 120))', 0) ),
(3, 'ST_Polygon', ST_Polygon('polygon((30 30, 25 35, 15 50,
35 80, 40 85, 80 90,70 75, 65 70, 55 50, 75 40, 60 30,
30 30))', 0) ),
(4, 'ST_MultiPoint', ST_MultiPoint('multipoint(20 20, 30 30,
20 40, 30 50)', 1))
```

La sentencia SELECT siguiente calcula la envoltura convexa de todas las geometrías creadas anteriormente y visualiza el resultado.

```
SELECT id, spatial_type, cast(geometry..ST_ConvexHull..ST_AsText
AS varchar(300)) AS convexhull
FROM sample_geometries
```

Resultados:

ID	SPATIAL_TYPE	CONVEXHULL
1	ST_LineString	POLYGON (( 20.00000000 40.00000000, 20.00000000 20.00000000, 30.00000000 30.00000000, 30.00000000 50.00000000, 20.00000000 40.00000000))
2	ST_Polygon	POLYGON (( 110.00000000 140.00000000, 110.00000000 120.00000000, 120.00000000 130.00000000, 110.00000000 140.00000000))
3	ST_Polygon	POLYGON (( 15.00000000 50.00000000, 25.00000000 35.00000000, 30.00000000 30.00000000, 60.00000000 30.00000000, 75.00000000 40.00000000, 80.00000000 90.00000000, 40.00000000 85.00000000, 35.00000000 80.00000000, 15.00000000 50.00000000))
4	ST_MultiPoint	POLYGON (( 20.00000000 40.00000000, 20.00000000 20.00000000, 30.00000000 30.00000000, 30.00000000 50.00000000, 20.00000000 40.00000000))

---

## ST\_CoordDim

ST\_CoordDim toma una geometría como parámetro de entrada y devuelve la dimensionalidad de sus coordenadas.

Si la geometría proporcionada no tiene coordenadas Z ni M, la dimensionalidad es 2. Si tiene coordenadas Z y no tiene M coordenadas, o si tiene coordenadas M y no

tiene coordenadas Z, la dimensionalidad es 3. Si tiene coordenadas Z y M, la dimensionalidad es 4. Si la geometría es un valor nulo, se vuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►—db2gse.ST\_CoordDim—(*—geometría—*)—►

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría de la que se debe recuperar la dimensionalidad.

## Tipo de retorno

INTEGER

## Ejemplo

Este ejemplo crea varias geometrías y luego determina la dimensionalidad de sus coordenadas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id CHARACTER(15), geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
  ('Empty Point', ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
  ('Linestring', ST_Geometry('linestring (10 10, 15 20)',0))

INSERT INTO sample_geoms VALUES
  ('Polygon', ST_Geometry('polygon((40 120, 90 120, 90 150,
  40 150, 40 120))' ,0))

INSERT INTO sample_geoms VALUES
  ('Multipoint M', ST_Geometry('multipoint m (10 10 5, 50 10
  6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
  ('Multipoint Z', ST_Geometry('multipoint z (47 34 295,
  23 45 678)' ,0))

INSERT INTO sample_geoms VALUES
  ('Point ZM', ST_Geometry('point zm (10 10 16 30)' ,0))

SELECT id, ST_CoordDim(geometry) COORDDIM
FROM sample_geoms
```

Resultados:

ID	COORDDIM
Empty Point	2
Linestring	2
Polygon	2
Multipoint M	3
Multipoint Z	3
Point ZM	4

---

## ST\_Crosses

ST\_Crosses toma dos geometrías como parámetros de entrada y devuelve un 1 si la primera geometría forma intersección con la segunda. En caso contrario, se devuelve un 0 (cero).

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si la primera geometría es un polígono o un multipolígono, o si la segunda geometría es un punto o un multipunto, o bien si cualquiera de la geometrías es un valor nulo, se devuelve un valor nulo. Si la dimensión de la geometría resultante de la intersección de las dos geometrías es una unidad menor que la dimensión máxima de las dos geometrías, y la geometría no es igual a ninguna de esas dos geometrías, se devuelve un 1. En otro caso, el resultado es 0 (cero).

### Sintaxis

►►db2gse.ST\_Crosses(—geometría1—,—geometría2—)◄◄

### Parámetro

#### geometría1

Un valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se debe verificar si se cruza con *geometría2*.

#### geometría2

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la que se debe comprobar si forma intersección con *geometría1*.

### Tipo de retorno

INTEGER

### Ejemplo

El código siguiente determina si las geometrías creadas forman intersección entre sí.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
  (1, ST_Geometry('polygon((30 30, 30 50, 50 50, 50 30, 30 30))' ,0))

INSERT INTO sample_geoms VALUES
  (2, ST_Geometry('linestring(40 50, 50 40)' ,0))

INSERT INTO sample_geoms VALUES
  (3, ST_Geometry('linestring(20 20, 60 60)' ,0))

SELECT a.id, b.id, ST_Crosses(a.geometry, b.geometry) Crosses
FROM sample_geoms a, sample_geoms b
```

Resultados:



ID	ID	CROSSES
1	1	-
2	1	0
3	1	1
1	2	-
2	2	0
3	2	1
1	3	-
2	3	1
3	3	0

## ST\_Difference

ST\_Difference toma dos geometrías como parámetros de entrada y devuelve la parte de la primera geometría que no forma intersección con la segunda.

Ambas geometrías deben tener la misma dirección. Si cualquiera de las geometrías tiene un valor nulo, se devuelve un valor nulo. Si la primera dimensión está vacía, se devuelve una dimensión vacía de tipo ST\_Point. Si la segunda geometría está vacía, se devuelve la primera geometría sin modificar.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_Difference—(—geometría1—,—geometría2—)—————►►

### Parámetro

#### geometría1

Valor de tipo ST\_Geometry que representa la primera geometría que se debe utilizar para calcular la diferencia con *geometría2*.

#### geometría2

Un valor de tipo ST\_Geometry que representa la segunda geometría que se utiliza para calcular la diferencia en *geometría1*.

#### Restricciones para datos geodésicos:

- Ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.
- ST\_Difference sólo soporta los tipos de datos ST\_Point, ST\_Polygon, ST\_MultiPoint y ST\_MultiPolygon.

### Tipo de retorno

db2gse.ST\_Geometry

La dimensión de la geometría devuelta es la misma que la de las geometrías de entrada.

## Ejemplos

En el ejemplo siguiente, los resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente crea y llena la tabla SAMPLE\_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
  (1, ST_Geometry('polygon((10 10, 10 20, 20 20, 20 10, 10 10))' ,0))

INSERT INTO sample_geoms VALUES
  (2, ST_Geometry('polygon((30 30, 30 50, 50 50, 50 30, 30 30))',0))

INSERT INTO sample_geoms VALUES
  (3, ST_Geometry('polygon((40 40, 40 60, 60 60, 60 40, 40 40))' ,0))

INSERT INTO sample_geoms VALUES
  (4, ST_Geometry('linestring(70 70, 80 80)' ,0))

INSERT INTO sample_geoms VALUES
  (5, ST_Geometry('linestring(75 75, 90 90)' ,0))
```

### Ejemplo 1

Este ejemplo busca la diferencia entre dos polígonos inconexos.

```
SELECT a.id, b.id, CAST(ST_AsText(ST_Difference(a.geometry, b.geometry))
  as VARCHAR(200)) Difference
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1 and b.id = 2
```

Resultados:

ID	ID	DIFFERENCE
1	2	POLYGON (( 10.00000000 10.00000000, 20.00000000 10.00000000, 20.00000000 20.00000000, 10.00000000 20.00000000, 10.00000000 10.00000000))

### Ejemplo 2

Este ejemplo busca la diferencia entre dos polígonos que forman una intersección.

```
SELECT a.id, b.id, CAST(ST_AsText(ST_Difference(a.geometry, b.geometry))
  as VARCHAR(200)) Difference
FROM sample_geoms a, sample_geoms b
WHERE a.id = 2 and b.id = 3
```

Resultados:

ID	ID	DIFFERENCE
2	3	POLYGON (( 30.00000000 30.00000000, 50.00000000 30.00000000, 50.00000000 40.00000000, 40.00000000 40.00000000, 40.00000000 50.00000000, 30.00000000 50.00000000, 30.00000000 30.00000000))

### Ejemplo 3

Este ejemplo busca la diferencia entre dos cadenas lineales que se solapan.

```
SELECT a.id, b.id, CAST(ST_AsText(ST_Difference(a.geometry, b.geometry))
as VARCHAR(100)) Difference
FROM sample_geoms a, sample_geoms b
WHERE a.id = 4 and b.id = 5
```

Resultados:

ID	ID	DIFFERENCE
4	5	LINESTRING ( 70.00000000 70.00000000, 75.00000000 75.00000000)

## ST\_Dimension

ST\_Dimension toma una geometría como parámetro de entrada y devuelve la dimensión del mismo.

Si la geometría proporcionada está vacía, se devuelve el valor -1. Para puntos y multipuntos, la dimensión es cero 0 (cero); para curvas y multicurvas, la dimensión es 1; y para polígonos y multipolígonos, la dimensión es 2. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►—db2gse.ST\_Dimension—(—geometría—)◄◄

### Parámetro

#### geometría

Valor de tipo ST\_Geometry que representa la geometría para la que se devuelve la dimensión.

### Tipo de retorno

INTEGER

### Ejemplo

Este ejemplo crea varias geometrías diferentes y determina sus dimensiones.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id char(15), geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
('Empty Point', ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
('Point ZM', ST_Geometry('point zm (10 10 16 30)',0))

INSERT INTO sample_geoms VALUES
('MultiPoint M', ST_Geometry('multipoint m (10 10 5,
50 10 6, 10 30 8)',0))

INSERT INTO sample_geoms VALUES
('LineString', ST_Geometry('linestring (10 10, 15 20)',0))

INSERT INTO sample_geoms VALUES
('Polygon', ST_Geometry('polygon((40 120, 90 120, 90 150,
```

```
40 150, 40 120))',0))
```

```
SELECT id, ST_Dimension(geometry) Dimension  
FROM sample_geoms
```

Resultados:

ID	DIMENSION
Empty Point	-1
Point ZM	0
MultiPoint M	0
LineString	1
Polygon	2

---

## ST\_Disjoint

ST\_Disjoint toma dos geometrías como parámetros de entrada y devuelve un 1 si dichas geometrías no forman intersección. Si las geometrías forman intersección, se devuelve un 0 (cero).

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las dos geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
db2gse.ST_Disjoint(geometría1,geometría2)
```

### Parámetro

#### *geometría1*

Valor de tipo ST\_Geometry que representa la geometría que se prueba si es inconexa respecto a *geometría2*.

#### *geometría2*

Valor de tipo ST\_Geometry que representa la geometría que se prueba si es inconexa respecto a *geometría1*.

### Tipo de retorno

INTEGER

### Ejemplos

#### Ejemplo 1

El código siguiente crea varias geometrías en la tabla SAMPLE\_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse  
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)
```

```
INSERT INTO sample_geoms VALUES  
(1, ST_Geometry('polygon((20 30, 30 30, 30 40, 20 40, 20 30))',0))
```

```

INSERT INTO sample_geoms VALUES
    (2, ST_Geometry('polygon((30 30, 30 50, 50 50, 50 30, 30 30))',0))

INSERT INTO sample_geoms VALUES
    (3, ST_Geometry('polygon((40 40, 40 60, 60 60, 60 40, 40 40))',0))

INSERT INTO sample_geoms VALUES
    (4, ST_Geometry('linestring(60 60, 70 70)' ,0))

INSERT INTO sample_geoms VALUES
    (5, ST_Geometry('linestring(30 30, 40 40)' ,0))

```

### Ejemplo 2

Este ejemplo determina si el primer polígono es inconexa respecto a cualquiera de las geometrías.

```

SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1

```

Resultados:

ID	ID	DISJOINT
1	1	0
1	2	0
1	3	1
1	4	1
1	5	0

### Ejemplo 3

Este ejemplo determina si el tercer polígono es inconexo respecto a cualquiera de las geometrías.

```

SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 3

```

Resultados:

ID	ID	DISJOINT
3	1	1
3	2	0
3	3	0
3	4	0
3	5	0

### Ejemplo 4

Este ejemplo determina si la segunda cadena lineal es inconexo respecto a cualquiera de las geometrías.

```

SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 5

```

Resultados:

ID	ID	DISJOINT
5	1	0

5	2	0
5	3	0
5	4	1
5	5	0

## ST\_Distance

ST\_Distance toma como parámetros de entrada dos geometrías y, opcionalmente, una unidad, y devuelve la distancia más corta entre cualquier punto de la primera geometría y cualquier punto de la segunda geometría, expresada en las unidades proporcionadas o en las unidades por omisión.

Para los datos geodésicos, ST\_Distance devuelve la *distancia geodésica* entre dos geometrías cualesquiera. La distancia geodésica es la distancia más corta en la superficie del elipsoide.

Si cualquiera de las geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

También puede llamar esta función como método cuando suministre una unidad de medida.

### Sintaxis

```
db2gse.ST_Distance(—geometría1—,—geometría2— [,—unidad—])
```

### Parámetro

#### geometría1

Valor de tipo ST\_Geometry que representa la geometría que se utiliza para calcular la distancia respecto a *geometría2*.

#### geometría2

Valor de tipo ST\_Geometry que representa la geometría que se utiliza para calcular la distancia respecto a *geometría1*.

#### unidad

Valor de tipo VARCHAR(128) que identifica las unidades utilizadas para medir el resultado. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST\_UNITS\_OF\_MEASURE.

Para los datos geodésicos, ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

Si se omite el parámetro *unidad*, se siguen las normas siguientes para determinar la unidad de medida utilizada para el resultado:

- Si *geometría1* está en un sistema de coordenadas proyectadas o geocéntricas, la unidad lineal asociada a este sistema de coordenadas es el valor por omisión.

- Si *geometría1* está en un sistema de coordenadas geográficas, pero no está en un SRS geodésico, la unidad angular asociada a este sistema de coordenadas es el valor por omisión.
- Si *geometría1* está en un SRS geodésico, la unidad por omisión de la medida son los metros.

**Restricciones para las conversiones de unidades:** Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La geometría está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La geometría está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La geometría está en un sistema de coordenadas geográficas, está en un SRS geodésico y se especifica una unidad lineal.

## Tipo de retorno

DOUBLE

## Ejemplos

### Ejemplo 1

Las siguientes sentencias de SQL crean y llenan las tablas SAMPLE\_GEOMETRIES1 y SAMPLE\_GEOMETRIES2.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_geometries1(id SMALLINT, spatial_type varchar(13),
  geometry ST_GEOMETRY)
```

```
CREATE TABLE sample_geometries2(id SMALLINT, spatial_type varchar(13),
  geometry ST_GEOMETRY)
```

```
INSERT INTO sample_geometries1(id, spatial_type, geometry)
VALUES
  ( 1, 'ST_Point', ST_Point('point(100 100)', 1) ),
  (10, 'ST_LineString', ST_LineString('linestring(125 125, 125 175)', 1) ),
  (20, 'ST_Polygon', ST_Polygon('polygon
    ((50 50, 50 150, 150 150, 150 50, 50 50))', 1) )
```

```
INSERT INTO sample_geometries2(id, spatial_type, geometry)
VALUES
  (101, 'ST_Point', ST_Point('point(200 200)', 1) ),
  (102, 'ST_Point', ST_Point('point(200 300)', 1) ),
  (103, 'ST_Point', ST_Point('point(200 0)', 1) ),
  (110, 'ST_LineString', ST_LineString('linestring(200 100, 200 200)', 1) ),
  (120, 'ST_Polygon', ST_Polygon('polygon
    ((200 0, 200 200, 300 200, 300 0, 200 0))', 1) )
```

### Ejemplo 2

La sentencia SELECT siguiente calcula la distancia entre las diversas geometrías contenidas en las tablas SAMPLE\_GEOMETRIES1 y SAMPLE\_GEOMETRIES2.

```
SELECT  sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
        sg2.id AS sg2_id, sg2.spatial_type AS sg2_type,
        cast(ST_Distance(sg1.geometry, sg2.geometry)
        AS Decimal(8, 4)) AS distance
FROM    sample_geometries1 sg1, sample_geometries2 sg2
ORDER BY sg1.id
```

Resultados:

SG1_ID	SG1_TYPE	SG1_ID	SG2_TYPE	DISTANCE
1	ST_Point	101	ST_Point	141.4213
1	ST_Point	102	ST_Point	223.6067
1	ST_Point	103	ST_Point	141.4213
1	ST_Point	110	ST_LineString	100.0000
1	ST_Point	120	ST_Polygon	100.0000
10	ST_LineString	101	ST_Point	79.0569
10	ST_LineString	102	ST_Point	145.7737
10	ST_LineString	103	ST_Point	145.7737
10	ST_LineString	110	ST_LineString	75.0000
10	ST_LineString	120	ST_Polygon	75.0000
20	ST_Polygon	101	ST_Point	70.7106
20	ST_Polygon	102	ST_Point	158.1138
20	ST_Polygon	103	ST_Point	70.7106
20	ST_Polygon	110	ST_LineString	50.0000
20	ST_Polygon	120	ST_Polygon	50.0000

### Ejemplo 3

La sentencia SELECT siguiente muestra cómo encontrar todas las geometrías que están dentro de una distancia de 100 la una respecto a las otras.

```
SELECT sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
       sg2.id AS sg2_id, sg2.spatial_type AS sg2_type,
       cast(ST_Distance(sg1.geometry, sg2.geometry)
          AS Decimal(8, 4)) AS distance
FROM   sample_geometries1 sg1, sample_geometries2 sg2
WHERE  ST_Distance(sg1.geometry, sg2.geometry) <= 100
```

Resultados:

SG1_ID	SG1_TYPE	SG1_ID	SG2_TYPE	DISTANCE
1	ST_Point	110	ST_LineString	100.0000
1	ST_Point	120	ST_Polygon	100.0000
10	ST_LineString	101	ST_Point	79.0569
10	ST_LineString	110	ST_LineString	75.0000
10	ST_LineString	120	ST_Polygon	75.0000
20	ST_Polygon	101	ST_Point	70.7106
20	ST_Polygon	103	ST_Point	70.7106
20	ST_Polygon	110	ST_LineString	50.0000
20	ST_Polygon	120	ST_Polygon	50.0000

### Ejemplo 4

La sentencia SELECT siguiente calcula la distancia en kilómetros entre las diversas geometrías.

Tablas SAMPLE\_GEOMETRIES1 y SAMPLE\_GEOMETRIES2.

```
SELECT sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
       sg2.id AS sg2_id, sg2.spatial_type AS sg2_type,
       cast(ST_Distance(sg1.geometry, sg2.geometry, 'KILOMETER')
          AS DECIMAL(10, 4)) AS distance
FROM   sample_geometries1 sg1, sample_geometries2 sg2
ORDER BY sg1.id
```

Resultados:

SG1_ID	SG1_TYPE	SG1_ID	SG2_TYPE	DISTANCE
1	ST_Point	101	ST_Point	12373.2168
1	ST_Point	102	ST_Point	16311.3816
1	ST_Point	103	ST_Point	9809.4713
1	ST_Point	110	ST_LineString	1707.4463



1 ST_Point	120 ST_Polygon	12373.2168
10 ST_LineString	101 ST_Point	8648.2333
10 ST_LineString	102 ST_Point	11317.3934
10 ST_LineString	103 ST_Point	10959.7313
10 ST_LineString	110 ST_LineString	3753.5862
10 ST_LineString	120 ST_Polygon	10891.1254
20 ST_Polygon	101 ST_Point	7700.5333
20 ST_Polygon	102 ST_Point	15039.8109
20 ST_Polygon	103 ST_Point	7284.8552
20 ST_Polygon	110 ST_LineString	6001.8407
20 ST_Polygon	120 ST_Polygon	14515.8872

---

## ST\_DistanceToPoint

ST\_DistanceToPoint toma una geometría de curva o multicurva y un geometría de punto como parámetros de entrada y devuelve la distancia junto con la geometría de curva en el punto especificado.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_DistanceToPoint—(—geometría\_curva—,—geometría\_punto—)————►◀

### Parámetro

#### geometría\_curva

Valor de tipo ST\_Curve o ST\_MultiCurve o uno de sus subtipos que representa la geometría que debe procesarse.

#### geometría\_punto

Valor de tipo ST\_Point que representa un punto en la curva especificada.

### Tipo de retorno

DOUBLE

### Ejemplo

#### Ejemplo 1

La sentencia de SQL siguiente crea la tabla SAMPLE\_GEOMETRIES con dos columnas. La columna de ID identifica exclusivamente cada fila. La columna GEOMETRY ST\_LineString almacena geometrías de ejemplo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

La sentencia de SQL siguiente inserta dos filas en la tabla SAMPLE\_GEOMETRIES.

```
INSERT INTO sample_geometries(id, geometry)
VALUES
(1,ST_LineString('LINESTRING ZM(0 0 0 0, 10 100 1000 10000)',1)),
(2,ST_LineString('LINESTRING ZM(10 100 1000 10000, 0 0 0 0)',1))
```

La sentencia SELECT siguiente y el conjunto de resultados correspondiente muestra cómo usar la función ST\_DistanceToPoint para buscar la distancia respecto al punto en la ubicación (1.5, 15.0).

```
SELECT ID, DECIMAL(ST_DistanceToPoint(geometry,ST_Point(1.5,15.0,1)),10,5)
AS DISTANCE FROM sample_geometries
```

ID	DISTANCE
1	15.07481
2	85.42394

2 registro(s) seleccionados.

---

## ST\_Edge\_GC\_USA

ST\_Edge\_GC\_USA es la función que hace uso del geocodificador DB2SE\_USA\_GEOCODER para geocodificar direcciones situadas en Estados Unidos en forma de puntos. Las direcciones se comparan con archivos EDGE, que se proporcionan en los datos de consulta del geocodificador disponibles para descargar.

Esta función toma como parámetros de entrada el nombre y el número de la calle, el nombre de la ciudad, el estado, el código postal y el identificador de sistema de referencia espacial del punto resultante y devuelve un valor de tipo ST\_Point. Además, se pueden especificar varios parámetros de configuración que afectan al proceso de geocodificación.

### Sintaxis

```
db2gse.ST_Edge_GC_USA(calle,ciudad,estado,código_postal,id_srs,
sensibilidad_ortográfica,grado_mínimo_coincidencia,desplaz_lateral,unidades_desplaz_lateral,desplaz_final,
mapa_base,archivo_localizador)
```

### Parámetro

**calle** Valor de tipo VARCHAR(128) que contiene el número y nombre de la calle correspondientes a la dirección que se debe geocodificar.

Este valor no debe ser nulo.

**ciudad**

Valor de tipo VARCHAR(128) que contiene el nombre de la ciudad correspondiente a la dirección que se debe geocodificar.

Este valor puede ser nulo si se especifica el parámetro *código postal*.

**estado** Valor de tipo VARCHAR(128) que contiene el nombre del estado correspondiente a la dirección que se debe geocodificar. El estado se puede especificar de forma abreviada o utilizando su nombre completo.

Este valor puede ser nulo si se especifica el parámetro *código postal*.

**código postal**

Valor de tipo VARCHAR(10) que contiene el código postal de la dirección que se debe geocodificar. El código postal se puede especificar en forma de 5 dígitos o utilizando la notación de 5+4.

Este valor puede ser nulo si se especifican los parámetros *ciudad* y *estado*.

**id\_srs** Valor de tipo INTEGER que contiene el identificador numérico del sistema de referencia espacial correspondiente al punto resultante. El valor debe identificar un sistema de referencia espacial existente, que hace uso de un sistema de coordenadas proyectadas basado en el sistema de coordenadas

geográficas GCS\_NORTH\_AMERICAN\_1983, o un sistema de referencia espacial que hace uso del propio sistema de coordenadas geográficas, GCS\_NORTH\_AMERICAN\_1983.

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

#### **sensibilidad\_ortográfica**

Valor de tipo INTEGER que especifica la sensibilidad ortográfica que se debe aplicar a la dirección proporcionada. El valor debe estar comprendido entre 0 (cero) y 100. Cuanto más alto sea este valor, más estricto será el geocodificador respecto a las diferencias en la ortografía de la dirección proporcionada. Las desviaciones detectadas dan como resultado una mayor penalización que se aplicará al grado final de coincidencia.

Si la sensibilidad ortográfica tiene un valor demasiado alto, probablemente se codificarán satisfactoriamente menos direcciones, y en su lugar se devolverá un valor nulo. Si la sensibilidad ortográfica tiene un valor demasiado bajo, pueden ser aceptadas como correctas más direcciones no apropiadas, debido al nivel aceptado para las diferencias ortográficas de las direcciones. **Recomendación:** Asigne el valor 60 a este parámetro.

Si este valor es nulo, la sensibilidad ortográfica se obtendrá a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, se utilizará una sensibilidad ortográfica igual a 60.

#### **grado\_mínimo\_coincidencia**

Valor de tipo INTEGER que contiene el grado mínimo de coincidencia que un punto debe tener para que sea considerado como coincidente con la dirección proporcionada. Este valor debe estar comprendido entre 0 (cero) y 100. Si el grado de coincidencia del punto es menor que *grado\_mínimo\_coincidencia*, se devuelve un valor nulo en lugar del punto, y la dirección no se geocodifica.

Diversos factores afectan al grado de coincidencia de un punto, tales como la calidad del mapa base, la sensibilidad ortográfica y la exactitud de la dirección. **Recomendación:** Asigne el valor 80 a este parámetro.

Si este valor es nulo, el grado mínimo de coincidencia se obtendrá a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, se utilizará un grado mínimo de coincidencia igual a 80.

#### **desplaz\_lateral**

Valor de tipo DOUBLE que especifica la distancia a la que debe colocarse un punto resultante respecto al centro de la calle. Este valor debe ser mayor o igual que 0 (cero). El parámetro *unidad\_desplaz\_lateral* identifica las unidades que se utilizan para medir el desplazamiento lateral.

Si este valor es nulo, el desplazamiento lateral se obtendrá a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, se utilizará un desplazamiento lateral igual a 0.0.

#### **unidades\_desplaz\_lateral**

Un valor de tipo VARCHAR(128) que contiene las unidades en las que se mide el parámetro *desplaz\_lateral*. El valor debe estar en una de estas unidades:

- Pulgadas
- Puntos
- Pies

- Yardas
- Millas
- Millas náuticas
- Milímetros
- Centímetros
- Metros
- Kilómetros
- Grados decimales
- Metros proyectados
- Unidades de datos de referencia

Si este valor es nulo, las unidades del desplazamiento lateral se obtendrán a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, el desplazamiento lateral se medirá en pies.

#### **desplaz\_final**

Valor de tipo INTEGER que indica la posición en un segmento de un punto que está exactamente al final de un segmento de calle. Este valor debe ser mayor o igual que 0 (cero). Este parámetro se utiliza para evitar colocar los puntos resultantes en el centro de una calle en las intersecciones. El desplazamiento final se mide en puntos (la resolución más alta posible) en el mapa base.

Si este valor es nulo, el desplazamiento final se obtendrá a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, se utilizará un desplazamiento final igual a 3.

#### **mapa\_base**

Valor de tipo VARCHAR(256) que contiene el nombre base y la vía de acceso totalmente calificada del archivo del mapa base (.edg). El geocodificador utiliza el archivo del mapa base para compararlo con las direcciones proporcionadas. Deben utilizarse los mapas base proporcionados por DB2 Spatial Extender. Puede utilizar este parámetro si ha colocado los mapas base en un directorio diferente.

Si este valor es nulo, la vía de acceso del mapa base se obtendrá a partir del archivo localizador. Si la vía de acceso no está especificada en el archivo localizador, el mapa base se buscará en el directorio sqllib de la instancia actual, en el directorio gse/refdata. El nombre de base del archivo buscado es usa.edg.

#### **archivo\_localizador**

Valor de tipo VARCHAR(256) que contiene el nombre base y la vía de acceso totalmente calificada del archivo localizador que contiene parámetros de configuración adicionales del geocodificador. Debe utilizarse el archivo localizador proporcionado por DB2 Spatial Extender.

Si este valor es nulo, el archivo localizador se buscará en el directorio sqllib de la instancia actual, en el subdirectorio gse/cfg/geocoder. El nombre base del archivo buscado es EDGELocator.loc.

### **Tipo de retorno**

db2gse.ST\_Point

## Ejemplos

### Ejemplo 1

El código siguiente crea una tabla `SAMPLE_GEOCODING` e inserta dos direcciones que luego se geocodifican. El grado mínimo de coincidencia se establece en 50 para las direcciones proporcionadas, y el sistema de referencia espacial de los puntos resultantes es 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geocoding (
  street VARCHAR(128),
  city VARCHAR(128),
  state VARCHAR(128),
  zip VARCHAR(5) )

INSERT INTO geocoding(street, city, state, zip)
VALUES ('1212 New York Ave NW', 'Washington', 'DC', '20005'),
('100 First North Street', 'San Jose', 'CA', NULL)

SELECT VARCHAR(ST_AsText(ST_Edge_GC_USA(street, city, state, zip, 1,
  CAST(NULL AS INTEGER), 50, CAST(NULL AS DOUBLE),
  CAST(NULL AS VARCHAR(128)), CAST(NULL AS INTEGER),
  CAST(NULL AS VARCHAR(256))), CAST(NULL AS VARCHAR(256)))), 50)
FROM sample_geocoding
```

Resultados:

```
1
-----
POINT ( -77.02829300 38.90049000)
POINT ( -121.94507200 37.28766700)
```

### Ejemplo 2

Este ejemplo crea un sistema de referencia espacial que hace uso de un sistema de coordenadas proyectadas. Para simplificar la interfaz de la función de geocodificación, se crea una función definida por el usuario para encapsular la función `ST_Edge_GC_USA`.

```
db2se create_srs <nombre_bd> -srsName CALIFORNIA -srsId 101 -xScale 1
-coordsysName NAD_1983_STATEPLANE_CALIFORNIA_I_FIPS_0401

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE FUNCTION California_GC (
  street VARCHAR(128), city VARCHAR(128), zip VARCHAR(10))
RETURNS db2gse.ST_Point
LANGUAGE SQL
RETURN db2gse.ST_Edge_GC_USA(street, city, 'CA', zip, 101,
  CAST(NULL AS INTEGER), CAST(NULL AS INTEGER),
  CAST(NULL AS DOUBLE), CAST(NULL AS VARCHAR(128)),
  CAST(NULL AS INTEGER), CAST(NULL AS VARCHAR(256)))

CREATE TABLE sample_geocoding (
  street VARCHAR(128),
  city VARCHAR(128),
  state VARCHAR(128),
  zip VARCHAR(5) )

INSERT INTO geocoding(street, city, state, zip)
VALUES ('100 First North Street', 'San Jose', 'CA', NULL)

SELECT VARCHAR(ST_AsText(California_GC(street, city, zip))), 50)
FROM sample_geocoding
```

Resultados:

```
1
-----
POINT ( 2004879.00000000 272723.00000000)
```

Los valores de las coordenadas X e Y de los puntos son diferentes a los del primer ejemplo debido a que se utiliza un sistema de referencia espacial diferente.

---

## ST\_Endpoint

ST\_Endpoint toma una curva como parámetro de entrada y devuelve el punto que es el último punto de la curva. El punto resultante se representa según el sistema de referencia espacial de la curva proporcionada.

Si la curva proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
►► db2gse.ST_EndPoint(—curva—) ◀◀
```

### Parámetro

**curva** Valor de tipo ST\_Curve que representa la geometría para la que se devuelve el último punto.

### Tipo de retorno

db2gse.ST\_Point

### Ejemplo

La sentencia SELECT siguiente determina el punto final de cada una de las geometrías de la tabla SAMPLE\_LINES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines(id INTEGER, line ST_LineString)
```

```
INSERT INTO sample_lines VALUES
(1, ST_LineString('linestring (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0) )
```

```
INSERT INTO sample_lines VALUES
(2, ST_LineString('linestring z (0 0 4, 5 5 5, 10 10 6, 5 5 7)', 0) )
```

```
SELECT id, CAST(ST_AsText(ST_EndPoint(line)) as VARCHAR(50)) Endpoint
FROM sample_lines
```

Resultados:

```
ID          ENDPOINT
-----
1 POINT ( 0.00000000 10.00000000)
2 POINT Z ( 5.00000000 5.00000000 7.00000000)
```

---

## ST\_Envelope

ST\_Envelope toma una geometría como parámetro de entrada y devuelve la envoltura de la geometría. La envoltura es un rectángulo que está representado por un polígono.

Si la geometría proporcionada es un punto, una cadena lineal horizontal o una cadena lineal vertical, se devuelve un rectángulo, que es ligeramente mayor que la geometría proporcionada. En otro caso, se devuelve como envoltura el rectángulo delimitador mínimo de la geometría. Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo. Para obtener el rectángulo delimitador mínimo exacto para todas las geometrías, utilice la función ST\_MBR.

Para los datos geodésicos, la envoltura es un polígono que delimita el círculo delimitador mínimo de la geometría.

También se puede invocar esta función como método.

### Sintaxis

► db2gse.ST\_Envelope(*—geometría—*) ◀

### Parámetro

#### geometría

Valor de tipo ST\_Geometry que representa la geometría para la que se devuelve la envoltura.

### Tipo de retorno

db2gse.ST\_Polygon

### Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea varias geometrías y luego determina sus envolturas. Para un punto y una cadena lineal (horizontal) que no están vacíos, la envoltura es un rectángulo que es ligeramente mayor que la geometría.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)
```

```
INSERT INTO sample_geoms VALUES
  (1, ST_Geometry('point EMPTY',0))
```

```
INSERT INTO sample_geoms VALUES
  (2, ST_Geometry('point zm (10 10 16 30)',0))
```

```
INSERT INTO sample_geoms VALUES
  (3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)',0))
```

```
INSERT INTO sample_geoms VALUES
  (4, ST_Geometry('linestring (10 10, 20 10)',0))
```

```
INSERT INTO sample_geoms VALUES
  (5, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))',0))
```

```
SELECT id, CAST(ST_AsText(ST_Envelope(geometry)) as VARCHAR(160)) Envelope
FROM sample_geoms
```

Resultados:

```
ID          ENVELOPE
-----
1 -
2 POLYGON (( 9.00000000 9.00000000, 11.00000000 9.00000000,
11.00000000 11.00000000, 9.00000000 11.00000000, 9.00000000 9.00000000))
3 POLYGON (( 10.00000000 10.00000000, 50.00000000 10.00000000,
50.00000000 30.00000000, 10.00000000 30.00000000, 10.00000000
10.00000000))
4 POLYGON (( 10.00000000 9.00000000, 20.00000000 9.00000000,
20.00000000 11.00000000, 10.00000000 11.00000000, 10.00000000
9.00000000))
5 POLYGON (( 40.00000000 120.00000000, 90.00000000 120.00000000,
90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
120.00000000))
```

---

## ST\_EnvIntersects

ST\_EnvIntersects toma dos geometrías como parámetros de entrada y devuelve un 1 si las envolturas de las dos geometrías forman intersección. En caso contrario, se devuelve un 0 (cero).

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

### Sintaxis

```
►►—db2gse.ST_EnvIntersects—(—geometría1—,—geometría2—)—◀◀
```

### Parámetro

#### geometría1

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría cuya envoltura se debe verificar si forma intersección con la envoltura de *geometría2*.

#### geometría2

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría cuya envoltura se debe verificar si forma intersección con la envoltura de *geometría1*.

### Tipo de retorno

INTEGER



## Ejemplo

Este ejemplo crea dos cadenas lineales paralelas y comprueba si forman intersección. Las cadenas lineales propiamente dichas no forman intersección, pero sí lo hacen sus envolturas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
  (1, ST_Geometry('linestring (10 10, 50 50)',0))

INSERT INTO sample_geoms VALUES
  (2, ST_Geometry('linestring (10 20, 50 60)',0))

SELECT a.id, b.id, ST_Intersects(a.geometry, b.geometry) Intersects,
       ST_EnvIntersects(a.geometry, b.geometry) Envelope_Intersects
FROM   sample_geoms a , sample_geoms b
WHERE  a.id = 1 and b.id=2
```

Resultados:

ID	ID	INTERSECTS	ENVELOPE_INTERSECTS
1	2	0	1

---

## ST\_EqualCoordsys

ST\_EqualCoordsys toma dos definiciones de sistemas de coordenadas como parámetros de entrada y devuelve el valor entero 1 (uno) si las definiciones proporcionadas son iguales. En otro caso, se devuelve el valor entero 0 (cero).

Las definiciones del sistema de coordenadas se comparan independientemente de las diferencias de espacios, paréntesis, caracteres en mayúsculas y minúsculas, y la representación de números de coma flotante.

Si cualquiera de las definiciones de sistemas de coordenadas tiene un valor nulo, se devuelve un valor nulo.

### Sintaxis

```
▶▶—db2gse.ST_EqualCoordsys—————▶▶
▶—(—sistema1_coordenadas—,—sistema2_coordenadas—)—————▶▶
```

### Parámetro

#### sistema1\_coordenadas

Valor de tipo VARCHAR(2048) que define el primer sistema de coordenadas que se debe comparar con *sistema2\_coordenadas*.

#### sistema2\_coordenadas

Valor de tipo VARCHAR(2048) que define el segundo sistema de coordenadas que se debe comparar con *sistema1\_coordenadas*.

### Tipo de retorno

INTEGER

## Ejemplo

Este ejemplo compara dos sistemas de coordenadas australianos para ver si son iguales.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
VALUES ST_EqualCoordSys(  
  (SELECT definition  
   FROM db2gse.ST_COORDINATE_SYSTEMS  
   WHERE coordsys_name='GCS_AUSTRALIAN') ,  
  
  (SELECT definition  
   FROM db2gse.ST_COORDINATE_SYSTEMS  
   WHERE coordsys_name='GCS_AUSTRALIAN_1984')  
)
```

Resultados:

```
1  
-----  
0
```

---

## ST\_Equals

ST\_Equals toma dos geometrías como parámetros de entrada y devuelve un 1 si las geometrías son iguales. En caso contrario, se devuelve un 0 (cero). El orden de los puntos utilizados para definir la geometría no es importante para comprobar la igualdad.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las dos geometrías proporcionadas es un valor nulo, se devuelve un valor nulo.

### Sintaxis

```
db2gse.ST_Equals(—geometría1—,—geometría2—)
```

### Parámetro

#### **geometría1**

Valor de tipo ST\_Geometry que representa la geometría que se va a comparar con la *geometría2*.

#### **geometría2**

Valor de tipo ST\_Geometry que representa la geometría que se va a comparar con la *geometría1*.

### Tipo de retorno

INTEGER

## Ejemplos

### Ejemplo 1

Este ejemplo crea dos polígonos que tienen sus coordenadas en un orden diferente. ST\_Equal muestra que estos polígonos son considerados como iguales.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
    (1, ST_Geometry('polygon((50 30, 30 30, 30 50, 50 50, 50 30))' ,0))

INSERT INTO sample_geoms VALUES
    (2, ST_Geometry('polygon((50 30, 50 50, 30 50, 30 30, 50 30))' ,0))

SELECT a.id, b.id, ST_Equals(a.geometry, b.geometry) Equals
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1 and b.id = 2
```

Resultados:

ID	ID	EQUALS
1	2	1

## Ejemplo 2

Este ejemplo crea dos geometrías con las mismas coordenadas X e Y, pero con coordenadas M (medidas) diferentes. Cuando las geometrías se comparan mediante la función ST\_Equal, se devuelve un 0 (cero) para indicar que las geometrías no son iguales.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
    (3, ST_Geometry('multipoint m(80 80 6, 90 90 7)' ,0))

INSERT INTO sample_geoms VALUES
    (4, ST_Geometry('multipoint m(80 80 6, 90 90 4)' ,0))

SELECT a.id, b.id, ST_Equals(a.geometry, b.geometry) Equals
FROM sample_geoms a, sample_geoms b
WHERE a.id = 3 and b.id = 4
```

Resultados:

ID	ID	EQUALS
3	4	0

## Ejemplo 3

Este ejemplo crea dos geometrías que tienen un conjunto diferente de coordenadas, pero que representan la misma geometría. ST\_Equal compara las geometrías e indica que son ciertamente iguales.

```
SET current function path = current function path, db2gse
CREATE TABLE sample_geoms ( id INTEGER, geometry ST_Geometry )

INSERT INTO sample_geoms VALUES
    (5, ST_LineString('linestring ( 10 10, 40 40 )', 0)),
    (6, ST_LineString('linestring ( 10 10, 20 20, 40 40)', 0))

SELECT a.id, b.id, ST_Equals(a.geometry, b.geometry) Equals
FROM sample_geoms a, sample_geoms b
WHERE a.id = 5 AND b.id = 6
```

Resultados:

ID	ID	EQUALS
-----	-----	-----
	5	6 1

## ST\_EqualSRS

ST\_EqualSRS toma dos identificadores de sistemas de referencia espacial como parámetros de entrada y devuelve un 1 si los sistemas de referencia espacial son iguales. En caso contrario, se devuelve un 0 (cero). Se comparan los desplazamientos, factores de escala y sistemas de coordenadas.

Si cualquiera de los identificadores de sistemas de coordenadas tiene un valor nulo, se devuelve un valor nulo.

### Sintaxis

```
►► db2gse.ST_EqualSRS(—id1_srs—,—id2_srs—)◄◄
```

### Parámetro

#### id1\_srs

Un valor de tipo INTEGER que identifica el primer sistema de referencia espacial que se comparará con el sistema de referencia espacial identificado por *id2\_srs*.

#### id2\_srs

Un valor de tipo INTEGER que identifica el segundo sistema de referencia espacial que se comparará con el sistema de referencia espacial identificado por *id1\_srs*.

### Tipo de retorno

INTEGER

### Ejemplo

Las siguientes llamadas a db2se crean dos sistemas de referencia espacial similares.

```
db2se create_srs SAMP_DB -srsId 12 -srsName NYE_12 -xOffset 0 -yOffset 0  
-xScale 1 -yScale 1 -coordsysName  
NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet
```

```
db2se create_srs SAMP_DB -srsId 22 -srsName NYE_22 -xOffset 0 -yOffset 0  
-xScale 1 -yScale 1 -coordsysName  
NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet
```

Estos sistemas de referencia espacial tienen los mismos valores de desplazamiento y escala, y hacen referencia a los mismos sistemas de coordenadas. La única diferencia está en el nombre definido y el ID de sistema de referencia espacial. Por tanto, la función de comparación devuelve un 1, que indica que los sistemas de referencia espacial son iguales.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
VALUES ST_EqualSRS(12, 22)
```

Resultados:

1  
-----  
1

---

## ST\_ExteriorRing

ST\_ExteriorRing toma un polígono como parámetro de entrada y devuelve su anillo exterior en forma de curva. La curva resultante se representa según el sistema de referencia espacial del polígono proporcionado.

Si el polígono proporcionado es un valor nulo o está vacío, se devuelve un valor nulo. Si el polígono no tiene anillos interiores, el anillo exterior devuelto es igual al perímetro del polígono.

También se puede invocar esta función como método.

### Sintaxis

►► db2gse.ST\_ExteriorRing(—polígono—) ◀◀

### Parámetro

#### polígono

Valor de tipo ST\_Polygon que representa el polígono para el que se debe devolver el anillo exterior.

### Tipo de retorno

db2gse.ST\_Curve

### Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea dos polígonos: uno con dos anillos interiores y otro sin anillos interiores, y luego determina sus anillos exteriores.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys VALUES
  (1, ST_Polygon('polygon((40 120, 90 120, 90 150, 40 150, 40 120),
    (50 130, 60 130, 60 140, 50 140, 50 130),
    (70 130, 80 130, 80 140, 70 140, 70 130))' ,0))

INSERT INTO sample_polys VALUES
  (2, ST_Polygon('polygon((10 10, 50 10, 10 30, 10 10))' ,0))

SELECT id, CAST(ST_AsText(ST_ExteriorRing(geometry))
  AS VARCHAR(180)) Exterior_Ring
FROM sample_polys
```

Resultados:

```

ID          EXTERIOR_RING
-----
1 LINESTRING ( 40.00000000 120.00000000, 90.00000000
120.00000000, 90.00000000 150.00000000, 40.00000000 150.00000000,
40.00000000 120.00000000)

2 LINESTRING ( 10.00000000 10.00000000, 50.00000000
10.00000000, 10.00000000 30.00000000, 10.00000000 10.00000000)

```

---

## ST\_FindMeasure o ST\_LocateAlong

ST\_FindMeasure o ST\_LocateAlong toma una geometría y una medida como parámetros de entrada y devuelve un multipunto o una multicurva que forma parte de la geometría proporcionada y cuya medida es exactamente la especificada.

Para los puntos y multipuntos, el resultado es todos los puntos que tienen la medida especificada. En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. Para las superficies y multisuperficies, el cálculo se realiza sobre el perímetro de la geometría.

Para los puntos y multipuntos, si no se encuentra la medida proporcionada, el resultado es una geometría vacía. Para todas las demás geometrías, si la medida proporcionada es menor que la medida más pequeña de la geometría o mayor que la medida más alta de la geometría, el resultado es una geometría vacía. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```

▶ db2gse.ST_FindMeasure (—geometría—, —medida—) ▶
  db2gse.ST_LocateAlong

```

### Parámetro

#### geometría

Un valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría en la que se buscan partes cuyas coordenadas M (medidas) contienen *medida*.

#### medida

Un valor de tipo DOUBLE que es la medida que las partes de *geometría* deben incluirse en el resultado.

### Tipo de retorno

db2gse.ST\_Geometry

### Ejemplos

#### Ejemplo 1

La sentencia CREATE TABLE siguiente crea la tabla SAMPLE\_GEOMETRIES. SAMPLE\_GEOMETRIES tiene dos columnas: la columna ID, que sirve para

identificar cada columna de forma unívoca, y la columna GEOMETRY de tipo ST\_Geometry, que contiene la geometría de ejemplo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_geometries(id SMALLINT, geometry ST_GEOMETRY)
```

Las sentencias INSERT siguientes insertan dos filas. La primera es una cadena lineal; la segunda es un multipunto.

```
INSERT INTO sample_geometries(id, geometry)
VALUES
  (1, ST_LineString('linestring m (2 2 3, 3 5 3, 3 3 6, 4 4 8)', 1)),
  (2, ST_MultiPoint('multipoint m
  (2 2 3, 3 5 3, 3 3 6, 4 4 6, 5 5 6, 6 6 8)', 1))
```

## Ejemplo 2

En la sentencia SELECT siguiente y en el resultado correspondiente, la función ST\_FindMeasure busca puntos cuya medida es 7. El resultado obtenido para la primera fila es un punto. En cambio, el resultado obtenido para la segunda fila es un punto vacío. En el caso de elementos geográficos lineales (geometrías con una dimensión mayor que 0), ST\_FindMeasure puede interpolar el punto; en cambio, para multipuntos, la medida resultante debe coincidir exactamente.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 7))
  AS varchar(45)) AS measure_7
FROM sample_geometries
```

Resultados:

```
ID      MEASURE_7
-----
1 POINT M ( 3.50000000 3.50000000 7.00000000)
2 POINT EMPTY
```

## Ejemplo 3

En la sentencia SELECT siguiente y en el resultado correspondiente, la función ST\_FindMeasure devuelve un punto y un multipunto. La medida resultante (6) coincide con las medidas existentes en las fuentes de datos de ST\_FindMeasure y del multipunto.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 6))
  AS varchar(120)) AS measure_6
FROM sample_geometries
```

Resultados:

```
ID      MEASURE_6
-----
1 POINT M ( 3.00000000 3.00000000 6.00000000)
2 MULTIPOINT M ( 3.00000000 3.00000000 6.00000000, 4.00000000
  4.00000000 6.00000000, 5.00000000 5.00000000 6.00000000)
```

---

## ST\_Generalize

ST\_Generalize toma una geometría y un umbral como parámetros de entrada y representa dicha geometría con un número reducido de puntos, al mismo tiempo que conserva las características generales de la geometría.

Se utiliza el algoritmo de simplificación de líneas de Douglas-Peucker, mediante el cual la secuencia de puntos que define la geometría se subdivide repetidamente

hasta que un tramo de los puntos se pueda sustituir por un segmento lineal recto. En este segmento lineal, ninguno de los puntos que lo definen se aparta del segmento lineal recto más allá del umbral especificado. Las coordenadas Z y M no se tienen en cuenta para la simplificación. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si la geometría proporcionada está vacía, se devuelve una geometría vacía de tipo ST\_Point. Si la geometría proporcionada o el umbral son un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►►—db2gse.ST\_Generalize—(—geometría—,—umbral—)—————►►

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la que se aplica la simplificación de línea.

### umbral

Valor de tipo DOUBLE que identifica el umbral que se utilizará para el algoritmo de simplificación de línea. El umbral debe ser mayor o igual a 0 (cero). Cuanto mayor sea el umbral, menor será el número de puntos que se utilizarán para representar la geometría generalizada. Para los datos geodésicos, la unidad utilizada para el umbral es el metro.

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplos

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

### Ejemplo 1

Se crea una cadena lineal con ocho puntos que van desde (10, 10) hasta (80, 80). El recorrido es casi una línea recta, pero algunos de los puntos están ligeramente fuera de la línea. Se puede utilizar la función ST\_Generalize para reducir el número de puntos de la línea.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_LineString)

INSERT INTO sample_lines VALUES
  (1, ST_LineString('linestring(10 10, 21 20, 34 26, 40 40,
                    52 50, 59 63, 70 71, 80 80)' ,0))
```

### Ejemplo 2

Cuando se utiliza un factor de generalización igual a 3, la cadena lineal queda reducida a cuatro coordenadas, y sigue siendo muy parecida a la representación original de la cadena lineal.



```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 3)) as VARCHAR(115))
       Generalize_3
FROM sample_lines
```

Resultados:

```
GENERALIZE 3
```

```
-----
LINESTRING ( 10.00000000 10.00000000, 34.00000000 26.00000000,
            59.00000000 63.00000000, 80.00000000 80.00000000)
```

### Ejemplo 3

Cuando se utiliza un factor de generalización igual a 6, la cadena lineal queda reducida a sólo dos coordenadas. Esto produce una cadena lineal más simple que el ejemplo anterior, pero se aparta más de la representación original.

```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 6)) as VARCHAR(65))
       Generalize_6
FROM sample_lines
```

Resultados:

```
GENERALIZE 6
```

```
-----
LINESTRING ( 10.00000000 10.00000000, 80.00000000 80.00000000)
```

---

## ST\_GeomCollection

Utilice `ST_GeomCollection` para construir una colección de geometrías.

`ST_GeomCollection` crea una colección de geometrías a partir de uno de estos datos de entrada siguientes:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma ESRI
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenida la colección de geometrías resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma ESRI o la representación GML es nula, se devuelve un valor nulo.

### Sintaxis

```
db2gse.ST_GeomCollection( ( wkt | wkb | forma | gml ) [, -id_srs ] )
```

### Parámetro

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la colección de geometrías resultante.

- wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la colección de geometrías resultante.
- forma** Valor de tipo BLOB(2G) que contiene la representación de forma ESRI de la colección de geometrías resultante.
- gml** Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), de la colección de geometrías resultante.
- id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la colección de geometrías resultante.

Si se omite el parámetro *id\_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_GeomCollection

## Notas

Si se omite el parámetro *id\_srs*, tal vez sea necesario convertir *wkt* y *gml* explícitamente al tipo de datos CLOB. De lo contrario, DB2 podría recurrir a la función utilizada para convertir valores desde el tipo de referencia REF(ST\_GeomCollection) al tipo ST\_GeomCollection. El ejemplo siguiente asegura que DB2 recurra a la función correcta:

## Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST\_GeomCollection para crear e insertar un multipunto, una multilinea y un multipolígono de una representación de texto convencional (WKT) y un multipunto de GML (Geographic Markup Language) en una columna GeomCollection.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_geomcollections(id INTEGER,
  geometry ST_GEOMCOLLECTION)
```

```
INSERT INTO sample_geomcollections(id, geometry)
VALUES
```

```
(4001, ST_GeomCollection('multipoint(1 2, 4 3, 5 6)', 1) ),
(4002, ST_GeomCollection('multilinestring(
  (33 2, 34 3, 35 6),
  (28 4, 29 5, 31 8, 43 12),
  (39 3, 37 4, 36 7))', 1) ),
(4003, ST_GeomCollection('multipolygon(((3 3, 4 6, 5 3, 3 3),
  (8 24, 9 25, 1 28, 8 24),
  (13 33, 7 36, 1 40, 10 43, 13 33)))', 1)),
(4004, ST_GeomCollection('<gml:MultiPoint srsName="EPSG:4269"
  ><gml:PointMember><gml:Point>
  <gml:coord><gml:X>10</gml:X>
  <gml:Y>20</gml:Y></gml: coord></gml:Point>
  </gml:PointMember><gml:PointMember>
  <gml:Point><gml:coord><gml:X>30</gml:X>
```

```
<gml:Y>40</gml:Y></gml:coord></gml:Point>
</gml:PointMember></gml:MultiPoint>', 1))
```

```
SELECT id, cast(geometry..ST_AsText AS varchar(350)) AS geomcollection
FROM sample_geomcollections
```

Resultados:

ID	GEOMCOLLECTION
4001	MULTIPOINT ( 1.00000000 2.00000000, 4.00000000 3.00000000, 5.00000000 6.00000000)
4002	MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000, 35.00000000 6.00000000), ( 28.00000000 4.00000000, 29.00000000 5.00000000, 31.00000000 8.00000000, 43.00000000 12.00000000), (39.00000000 3.00000000, 37.00000000 4.00000000, 36.00000000 7.00000000))
4003	MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000, 1.00000000 40.00000000, 7.00000000 36.00000000, 13.00000000 33.00000000)), (( 8.00000000 24.00000000, 9.00000000 25.00000000, 1.00000000 28.00000000, 8.00000000 24.00000000)), (( 3.00000000 3.00000000, 5.00000000 3.00000000, 4.00000000 6.00000000, 3.00000000 3.00000000)))
4004	MULTIPOINT ( 10.00000000 20.00000000, 30.00000000 40.00000000)

## ST\_GeomCollFromTxt

ST\_GeomCollFromTxt toma como parámetros de entrada una representación de texto convencional de una colección de geometrías y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la colección de geometrías correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_GeomCollection. Es recomendable debido a su flexibilidad: ST\_GeomCollection acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

### Sintaxis

```
db2gse.ST_GeomCollFromTxt(—wkt—, —id_srs—)
```

### Parámetro

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la colección de geometrías resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la colección de geometrías resultante.

Si se omite el parámetro *id\_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_GeomCollection

## Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST\_GeomCollFromTxt para crear e insertar un multipunto, una multilinea y un multipolígono de una representación de texto convencional (WKT) en una columna GeomCollection.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geomcollections(id INTEGER, geometry ST_GEOMCOLLECTION)

INSERT INTO sample_geomcollections(id, geometry)
VALUES
  (4011, ST_GeomCollFromTxt('multipoint(1 2, 4 3, 5 6)', 1) ),
  (4012, ST_GeomCollFromTxt('multilinestring(
    (33 2, 34 3, 35 6),
    (28 4, 29 5, 31 8, 43 12),
    (39 3, 37 4, 36 7))', 1) ),
  (4013, ST_GeomCollFromTxt('multipolygon(((3 3, 4 6, 5 3, 3 3),
    (8 24, 9 25, 1 28, 8 24),
    (13 33, 7 36, 1 40, 10 43, 13 33)))', 1))

SELECT id, cast(geometry..ST_AsText AS varchar(340))
       AS geomcollection
FROM   sample_geomcollections
```

Resultados:

ID	GEOMCOLLECTION
4011	MULTIPOINT ( 1.00000000 2.00000000, 4.00000000 3.00000000, 5.00000000 6.00000000)
4012	MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000, 35.00000000 6.00000000),( 28.00000000 4.00000000, 29.00000000 5.00000000, 31.00000000 8.00000000, 43.00000000 12.00000000),( 39.00000000 3.00000000, 37.00000000 4.00000000, 36.00000000 7.00000000))
4013	MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000, 1.00000000 40.00000000, 7.00000000 36.00000000, 13.00000000 33.00000000)), (( 8.00000000 24.00000000, 9.00000000 25.00000000, 1.00000000 28.00000000, 8.00000000 24.00000000)),(( 3.00000000 3.00000000, 5.00000000 3.00000000, 4.00000000 6.00000000, 3.00000000 3.00000000)))

---

## ST\_GeomCollFromWKB

ST\_GeomCollFromWKB toma como parámetros de entrada una representación binaria convencional de una colección de geometrías y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la colección de geometrías correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST\_GeomCollection.

## Sintaxis

► db2gse.ST\_GeomCollFromTxt(*wkb*, *id\_srs*)

## Parámetro

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la colección de geometrías resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la colección de geometrías resultante.

Si se omite el parámetro *id\_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_GeomCollection

## Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST\_GeomCollFromWKB para crear y consultar las coordenadas de una colección de geometrías en una representación binaria convencional. Las filas se insertan en la tabla SAMPLE\_GEOMCOLLECTION con los ID 4021 y 4022, y las colecciones de geometrías en el sistema de referencia espacial 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_geomcollections(id INTEGER,  
  geometry ST_GEOMCOLLECTION, wkb BLOB(32k))
```

```
INSERT INTO sample_geomcollections(id, geometry)  
VALUES  
  (4021, ST_GeomCollFromTxt('multipoint(1 2, 4 3, 5 6)', 1)),  
  (4022, ST_GeomCollFromTxt('multilinestring(  
    (33 2, 34 3, 35 6),  
    (28 4, 29 5, 31 8, 43 12))', 1))
```

```
UPDATE sample_geomcollections AS temp_correlated  
SET wkb = geometry..ST_AsBinary  
WHERE id = temp_correlated.id
```

```
SELECT id, cast(ST_GeomCollFromWKB(wkb)..ST_AsText  
  AS varchar(190)) AS GeomCollection  
FROM sample_geomcollections
```

Resultados:

```
ID          GEOMCOLLECTION
-----
          4021 MULTIPOINT ( 1.00000000 2.00000000, 4.00000000
          3.00000000, 5.00000000 6.00000000)

          4022 MULTILINESTRING (( 33.00000000 2.00000000,
          34.00000000 3.00000000, 35.00000000 6.00000000),( 28.00000000
          4.00000000, 29.00000000 5.00000000, 31.00000000 8.00000000,
          43.00000000 12.00000000))
```

## ST\_Geometry

ST\_Geometry crea una geometría a partir de uno de estos datos de entrada siguientes:

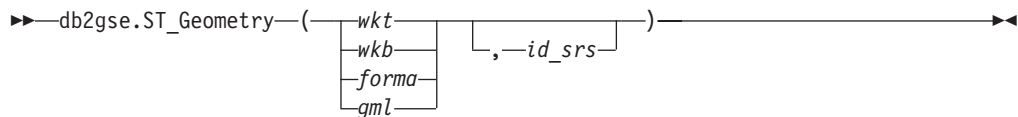
- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma ESRI
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenida la geometría resultante.

El tipo dinámico de la geometría resultante es uno de los subtipos de ST\_Geometry de los que pueden crearse instancias.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma ESRI o la representación GML es nula, se devuelve un valor nulo.

### Sintaxis



### Parámetro

- wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la geometría resultante.
- wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la geometría resultante.
- forma** Valor de tipo BLOB(2G) que contiene la representación de forma ESRI de la geometría resultante.
- gml** Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), de la geometría resultante.
- id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

Si se omite el parámetro *id\_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST\_Geometry para crear e insertar un punto desde una representación de texto convencional de un punto o una línea desde una representación GML (Geographic Markup Language) de una línea.

La función ST\_Geometry es la más flexible de las funciones constructoras de tipos espaciales, pues puede crear cualquier tipo espacial a partir de diversas representaciones de geometrías. ST\_LineFromText sólo puede crear una línea a partir de una representación WKT de una línea. ST\_WKTTToSql puede crear cualquier tipo, pero sólo a partir de una representación WKT.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id INTEGER, geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, geometry)
VALUES
  (7001, ST_Geometry('point(1 2)', 1) ),
  (7002, ST_Geometry('linestring(33 2, 34 3, 35 6)', 1) ),
  (7003, ST_Geometry('polygon((3 3, 4 6, 5 3, 3 3))', 1)),
  (7004, ST_Geometry('<gml:Point srsName="";EPSG:4269";><gml:coord>
  <gml:X>50</gml:X><gml:Y>60</gml:Y></gml:coord>
  </gml:Point>', 1))

SELECT id, cast(geometry..ST_AsText AS varchar(120)) AS geometry
FROM sample_geometries
```

Resultados:

ID	GEOMETRY
7001	POINT ( 1.00000000 2.00000000)
7002	LINestring ( 33.00000000 2.00000000, 34.00000000 3.00000000, 35.00000000 6.00000000)
7003	POLYGON (( 3.00000000 3.00000000, 5.00000000 3.00000000, 4.00000000 6.00000000, 3.00000000 3.00000000))
7004	POINT ( 50.00000000 60.00000000)

---

## ST\_GeometryN

ST\_GeometryN toma una colección de geometrías y un índice como parámetros de entrada y devuelve la geometría de la colección que está identificada por el índice. La geometría resultante se representa según el sistema de referencia espacial de la colección de geometrías proporcionada.

Si la colección de geometrías proporcionada tiene un valor nulo o está vacío, o si el índice es menor que 1 o mayor que el número de geometrías de la colección, se devuelve un valor nulo y se emite una condición de aviso (01HS0).

También se puede invocar esta función como método.

## Sintaxis

►► db2gse.ST\_GeometryN(—colección—,—índice—)◄◄

## Parámetro

### colección

Valor de tipo ST\_GeomCollection o uno de sus subtipos que representa la colección de geometrías dentro de la cual se debe localizar la geometría que ocupa la posición *n*.

**índice** Un valor de tipo INTEGER que identifica la geometría número *n* que se debe obtener desde *colección*.

Si *índice* es menor que 1 o mayor que el número de geometrías de la colección, se devuelve un valor nulo y se emite un aviso (SQLSTATE 01HS0).

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplo

El código siguiente muestra cómo seleccionar la segunda geometría dentro de la colección de geometrías.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_geomcollections (id INTEGER,
  geometry ST_GEOMCOLLECTION)
```

```
INSERT INTO sample_geomcollections(id, geometry)
VALUES
```

```
  (4001, ST_GeomCollection('multipoint(1 2, 4 3)', 1) ),
  (4002, ST_GeomCollection('multilinestring(
    (33 2, 34 3, 35 6),
    (28 4, 29 5, 31 8, 43 12),
    (39 3, 37 4, 36 7))', 1) ),
  (4003, ST_GeomCollection('multipolygon(((3 3, 4 6, 5 3, 3 3),
    (8 24, 9 25, 1 28, 8 24),
    (13 33, 7 36, 1 40, 10 43, 13 33)))', 1))
```

```
SELECT id, cast(ST_GeometryN(geometry, 2)..ST_AsText AS varchar(110))
  AS second_geometry
FROM   sample_geomcollections
```

Resultados:

```
ID          SECOND_GEOMETRY
-----
4001 POINT ( 4.00000000 3.00000000)

4002 LINESTRING ( 28.00000000 4.00000000, 29.00000000 5.00000000,
31.00000000 8.00000000, 43.00000000 12.00000000)
```



```
4003 POLYGON (( 8.00000000 24.00000000, 9.00000000 25.00000000,
1.00000000 28.00000000, 8.00000000 24.00000000))
```

---

## ST\_GeometryType

ST\_GeometryType toma una geometría como parámetro de entrada y devuelve el nombre de tipo calificado al completo del tipo dinámico de dicha geometría.

Las funciones TYPE\_SCHEMA y TYPE\_NAME de DB2 tienen el mismo efecto.

También se puede invocar esta función como método.

### Sintaxis

```
►►—db2gse.ST_GeometryType—(—geometría—)—————►
```

### Parámetro

#### geometría

Valor de tipo ST\_Geometry para el que se devolverá el tipo de geometría.

### Tipo de retorno

VARCHAR(128)

### Ejemplos

El código siguiente muestra cómo determinar el tipo de una geometría.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_geometries (id INTEGER, geometry ST_GEOMETRY)
```

```
INSERT INTO sample_geometries(id, geometry)
VALUES
(7101, ST_Geometry('point(1 2)', 1) ),
(7102, ST_Geometry('linestring(33 2, 34 3, 35 6)', 1) ),
(7103, ST_Geometry('polygon((3 3, 4 6, 5 3, 3 3))', 1)),
(7104, ST_Geometry('multipoint(1 2, 4 3)', 1) )
```

```
SELECT id, geometry..ST_GeometryType AS geometry_type
FROM sample_geometries
```

Resultados:

ID	GEOMETRY_TYPE
7101	"DB2GSE"."ST_POINT"
7102	"DB2GSE"."ST_LINestring"
7103	"DB2GSE"."ST_POLYGON"
7104	"DB2GSE"."ST_MULTIPPOINT"

---

## ST\_GeomFromText

ST\_GeomFromText toma como parámetros de entrada una representación de texto convencional de una geometría y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la geometría correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST\_Geometry.

## Sintaxis

```
db2gse.ST_GeomFromText(wkt, id_srs)
```

## Parámetro

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la geometría resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

Si se omite el parámetro *id\_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

En este ejemplo, se utiliza la función ST\_GeomFromText para crear e insertar un punto a partir de su representación de texto convencional (WKT).

El código siguiente inserta filas en la tabla SAMPLE\_POINTS con identificadores y geometrías en el sistema de referencia espacial 1 utilizando la representación WKT.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id INTEGER, geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, geometry)
VALUES
  (1251, ST_GeomFromText('point(1 2)', 1) ),
  (1252, ST_GeomFromText('linestring(33 2, 34 3, 35 6)', 1) ),
  (1253, ST_GeomFromText('polygon((3 3, 4 6, 5 3, 3 3))', 1))
```

La sentencia SELECT siguiente obtiene el ID y las geometrías a partir de la tabla SAMPLE\_GEOMETRIES.

```
SELECT id, cast(geometry..ST_AsText AS varchar(105))
       AS geometry
FROM sample_geometries
```

Resultados:

ID	GEOMETRY
1251	POINT ( 1.00000000 2.00000000)
1252	LINESTRING ( 33.00000000 2.00000000, 34.00000000 3.00000000, 35.00000000 6.00000000)
1253	POLYGON (( 3.00000000 3.00000000, 5.00000000 3.00000000, 4.00000000 6.00000000, 3.00000000 3.00000000))

## ST\_GeomFromWKB

ST\_GeomFromWKB toma como parámetros de entrada una representación de texto convencional de una geometría y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la geometría correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST\_Geometry.

### Sintaxis

```
db2gse.ST_GeomFromWKB(wkb [, id_srs])
```

### Parámetro

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la geometría resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

Si se omite el parámetro *id\_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si el parámetro *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_Geometry

### Ejemplos

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST\_GeomFromWKB para crear e insertar una línea a partir de una representación binaria convencional (WKB) de una línea.

El ejemplo siguiente inserta un registro en la tabla SAMPLE\_GEOMETRIES, con un ID y una geometría en el sistema de referencia espacial 1, en una representación WKB.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries (id INTEGER, geometry ST_GEOMETRY,
    wkb BLOB(32K))

INSERT INTO sample_geometries(id, geometry)
VALUES
    (1901, ST_GeomFromText('point(1 2)', 1) ),
    (1902, ST_GeomFromText('linestring(33 2, 34 3, 35 6)', 1) ),
    (1903, ST_GeomFromText('polygon((3 3, 4 6, 5 3, 3 3))', 1))

UPDATE sample_geometries AS temp_correlated
SET wkb = geometry..ST_AsBinary
WHERE id = temp_correlated.id

SELECT id, cast(ST_GeomFromWKB(wkb)..ST_AsText AS varchar(190))
    AS geometry
FROM sample_geometries
```

Resultados:

ID	GEOMETRY
1901	POINT ( 1.00000000 2.00000000)
1902	LINestring ( 33.00000000 2.00000000, 34.00000000 3.00000000, 35.00000000 6.00000000)
1903	POLYGON (( 3.00000000 3.00000000, 5.00000000 3.00000000, 4.00000000 6.00000000, 3.00000000 3.00000000))

## ST\_GetIndexParms

ST\_GetIndexParms toma como parámetro de entrada el identificador de un índice espacial o columna espacial y devuelve los parámetros utilizados para definir el índice en la columna espacial. Opcionalmente se puede especificar un número de parámetro, en cuyo caso sólo se obtiene el tamaño de retícula identificado por el número.

### Sintaxis

```
db2gse.ST_GetIndexParms(
    esquema_índice, nombre_índice
    esquema_tabla, nombre_tabla, nombre_columna
    , número_tamaño_retícula)
```

### Parámetro

#### esquema\_índice

Valor de tipo VARCHAR(128) que identifica el esquema donde está contenido el índice espacial cuyo nombre no calificado es *nombre\_índice*. El nombre de esquema distingue entre mayúsculas y minúsculas, y debe aparecer listado en la vista de catálogo SYSCAT.SCHEMATA.

Si este parámetro tiene un valor nulo, se utiliza el valor del registro especial CURRENT SCHEMA como nombre de esquema para el índice espacial.

**nombre\_índice**

Valor de tipo VARCHAR(128) que contiene el nombre no calificado del índice espacial para el cual se obtienen los parámetros del índice. El nombre de índice distingue entre mayúsculas y minúsculas, y debe aparecer listado en la vista de catálogo SYSCAT.INDEXES para el esquema *esquema\_índice*.

**esquema\_tabla**

Valor de tipo VARCHAR(128) que identifica el esquema donde está contenida la tabla cuyo nombre no calificado es *nombre\_tabla*. El nombre de esquema distingue entre mayúsculas y minúsculas, y debe aparecer listado en la vista de catálogo SYSCAT.SCHEMATA.

Si este parámetro tiene un valor nulo, se utiliza el valor del registro especial CURRENT SCHEMA como nombre de esquema para el índice espacial.

**nombre\_tabla**

Valor de tipo VARCHAR(128) que contiene el nombre no calificado de la tabla que tiene la columna espacial *nombre\_columna*. El nombre de tabla distingue entre mayúsculas y minúsculas y debe aparecer listado en la vista de catálogo SYSCAT.TABLES para el esquema *esquema\_tabla*.

**nombre\_columna**

Un valor de tipo VARCHAR(128) que identifica la columna de la tabla *esquema\_tabla.nombre\_tabla* para la cual se obtienen los parámetros de índice del índice espacial definido en esa columna. El nombre de columna distingue entre mayúsculas y minúsculas, y debe aparecer listado en la vista de catálogo SYSCAT.COLUMNS para la tabla *esquema\_tabla.nombre\_tabla*.

Si no existe ningún índice espacial definido en la columna, se emite un error (SQLSTATE 38SQ0).

**número\_tamaño\_retícula**

Valor de tipo DOUBLE que identifica el parámetro para el cual se obtiene su valor o valores.

Si este valor es menor que 1 o mayor que 3, se emite un error (SQLSTATE 38SQ1).

**Tipo de retorno**

DOUBLE (si está especificado *número\_tamaño\_retícula*)

Si *número\_tamaño\_retícula* no está especificado, se devuelve una tabla que contiene dos columnas: ORDINAL y VALUE. La columna ORDINAL es de tipo INTEGER, y la columna VALUE es de tipo DOUBLE.

Si los parámetros se obtienen para un índice reticular, la columna ORDINAL contiene los valores 1, 2 y 3 para el primer, segundo y tercer tamaño de retícula, respectivamente. La columna VALUE contiene los tamaños de retícula.

La columna VALUE contiene los valores respectivos para cada parámetro.

## Ejemplos

### Ejemplo 1

Este código crea una tabla con una columna espacial y un índice espacial.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sch.offices (name VARCHAR(30), location ST_Point )

CREATE INDEX sch.idx ON sch.offices(location)
EXTEND USING db2gse.spatial_index(1e0, 10e0, 1000e0)
```

Se puede utilizar la función ST\_GetIndexParms para obtener los valores de los parámetros que se utilizaron al crear el índice espacial.

### Ejemplo 2

Este ejemplo muestra cómo obtener por separado los tres tamaños de retícula para un índice reticular espacial, mediante la especificación explícita del parámetro a devolver, identificado por su número.

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 1)
```

Resultados:

```
1
-----
+1.000000000000000E+000
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 2)
```

Resultados:

```
1
-----
+1.000000000000000E+001
VALUES ST_GetIndexParms('SCH', 'IDX', 3)
```

Resultados:

```
1
-----
+1.000000000000000E+003
```

### Ejemplo 3

Este ejemplo muestra cómo obtener todos los parámetros de un índice reticular espacial. La función ST\_GetIndexParms obtiene una tabla que indica el número de parámetro y el correspondiente tamaño de retícula.

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION') ) AS t
```

Resultados:

```
ORDINAL    VALUE
-----
1          +1.000000000000000E+000
2          +1.000000000000000E+001
3          +1.000000000000000E+003
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'IDX') ) AS t
```

Resultados:

ORDINAL	VALUE
1	+1.000000000000000E+000
2	+1.000000000000000E+001
3	+1.000000000000000E+003

## ST\_InteriorRingN

ST\_InteriorRingN toma un polígono y un índice como parámetros de entrada y devuelve el anillo interior identificado por dicho índice en forma de cadena lineal. Los anillos interiores se organizan según las normas definidas por las rutinas definidas por las rutinas de verificación de la geometría interna.

Si el polígono proporcionado es un valor nulo o está vacío, o si no tiene anillos interiores, se devuelve un valor nulo. Si el índice es menor que 1 o mayor que el número de anillos interiores del polígono, se devuelve un valor nulo y se emite una condición de aviso (1HS1).

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_InteriorRingN—(—*polígono*—,—*índice*—)—————►►

### Parámetro

#### polígono

Valor de tipo ST\_Polygon que representa la geometría a partir de la cual se devuelve el anillo interior identificado por *índice*.

**índice** Un valor de tipo INTEGER que identifica el anillo interior *n* que se devolverá. Si no hay ningún anillo interior identificado por *índice*, se emite una condición de aviso (01HS1).

### Tipo de retorno

db2gse.ST\_Curve

### Ejemplo

El ejemplo siguiente crea un polígono con dos anillos interiores. A continuación, se invoca a ST\_InteriorRingN para obtener el segundo anillo interior.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys VALUES
  (1, ST_Polygon('polygon((40 120, 90 120, 90 150, 40 150, 40 120),
    (50 130, 60 130, 60 140, 50 140, 50 130),
    (70 130, 80 130, 80 140, 70 140, 70 130))' ,0))

SELECT id, CAST(ST_AsText(ST_InteriorRingN(geometry, 2)) as VARCHAR(180))
  Interior_Ring
FROM sample_polys
```

Resultados:

```
ID          INTERIOR_RING
-----
1 LINESTRING ( 70.00000000 130.00000000, 70.00000000 140.00000000,
80.00000000 140.00000000, 80.00000000 130.00000000, 70.00000000 130.00000000)
```

## ST\_Intersection

ST\_Intersection toma dos geometrías como parámetros de entrada y devuelve la geometría que es la intersección de estas dos geometrías. La intersección es la parte común de la primera geometría y de la segunda. La geometría resultante se representa según el sistema de referencia espacial de la primera geometría.

Si es posible, el tipo específico de la geometría devuelta será ST\_Point, ST\_LineString o ST\_Polygon. Por ejemplo, la intersección de un punto y un polígono es vacía o es un punto único, que se representa como ST\_MultiPoint.

Si cualquiera de las dos geometrías es un valor nulo, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

También se puede invocar esta función como método.

### Sintaxis

```
►►—db2gse.ST_Intersection—(—geometría1—,—geometría2—)—————►►
```

### Parámetro

#### geometría1

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la primera geometría para calcular la intersección con *geometría2*.

#### geometría2

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la segunda geometría para calcular la intersección con *geometría1*.

Para los datos geodésicos, ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

### Tipo de retorno

db2gse.ST\_Geometry

La dimensión de la geometría devuelta es la de la entrada con la dimensión inferior, con la excepción de las cadenas lineales de los datos geodésicos. Para los datos geodésicos, la dimensión de la intersección de dos cadenas lineales es 0 (en otras palabras, la intersección es un punto o un multipunto).

### Ejemplo

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.



Este ejemplo crea varias geometrías diferentes y luego determina la intersección (si existe) con la primera geometría.

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
    (1, ST_Geometry('polygon((30 30, 30 50, 50 50, 50 30, 30 30))' ,0))

INSERT INTO sample_geoms VALUES
    (2, ST_Geometry('polygon((20 30, 30 30, 30 40, 20 40, 20 30))' ,0))

INSERT INTO sample_geoms VALUES
    (3, ST_Geometry('polygon((40 40, 40 60, 60 60, 60 40, 40 40))' ,0))

INSERT INTO sample_geoms VALUES
    (4, ST_Geometry('linestring(60 60, 70 70)' ,0))

INSERT INTO sample_geoms VALUES
    (5, ST_Geometry('linestring(30 30, 60 60)' ,0))

SELECT a.id, b.id, CAST(ST_AsText(ST_Intersection(a.geometry, b.geometry))
    as VARCHAR(150)) Intersection
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1

```

Resultados:

ID	ID	INTERSECTION
1	1	POLYGON (( 30.00000000 30.00000000, 50.00000000 30.00000000, 50.00000000 50.00000000, 30.00000000 50.00000000, 30.00000000 30.00000000))
1	2	LINestring ( 30.00000000 40.00000000, 30.00000000 30.00000000)
1	3	POLYGON (( 40.00000000 40.00000000, 50.00000000 40.00000000, 50.00000000 50.00000000, 40.00000000 50.00000000, 40.00000000 40.00000000))
1	4	POINT EMPTY
1	5	LINestring ( 30.00000000 30.00000000, 50.00000000 50.00000000)

5 registro(s) seleccionados.

---

## ST\_Intersects

ST\_Intersects toma dos geometrías como parámetros de entrada y devuelve un 1 si dichas geometrías forman intersección. Si las geometrías no forman intersección, se devuelve un 0 (cero).

Si cualquiera de las geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

## Sintaxis

►► db2gse.ST\_Intersects(—*geometría1*—,—*geometría2*—)◀◀

## Parámetro

### **geometría1**

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para probar la intersección con *geometría2*.

### **geometría2**

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para probar la intersección con *geometría1*.

**Restricciones:** Para los datos geodésicos, ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

## Tipo de retorno

INTEGER

## Ejemplo

Las siguientes sentencias crean y llenan las tablas SAMPLE\_GEOMETRIES1 y SAMPLE\_GEOMETRIES2.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries1(id SMALLINT, spatial_type varchar(13),
    geometry ST_GEOMETRY);
CREATE TABLE sample_geometries2(id SMALLINT, spatial_type varchar(13),
    geometry ST_GEOMETRY);

INSERT INTO sample_geometries1(id, spatial_type, geometry)
VALUES
    ( 1, 'ST_Point', ST_Point('point(550 150)', 1) ),
    (10, 'ST_LineString', ST_LineString('linestring(800 800, 900 800)', 1)),
    (20, 'ST_Polygon', ST_Polygon('polygon((500 100, 500 200, 700 200,
        700 100, 500 100))', 1) )

INSERT INTO sample_geometries2(id, spatial_type, geometry)
VALUES
    (101, 'ST_Point', ST_Point('point(550 150)', 1) ),
    (102, 'ST_Point', ST_Point('point(650 200)', 1) ),
    (103, 'ST_Point', ST_Point('point(800 800)', 1) ),
    (110, 'ST_LineString', ST_LineString('linestring(850 250, 850 850)', 1)),
    (120, 'ST_Polygon', ST_Polygon('polygon((650 50, 650 150, 800 150,
        800 50, 650 50))', 1)),
    (121, 'ST_Polygon', ST_Polygon('polygon((20 20, 20 40, 40 40, 40 20,
        20 20))', 1) )
```

La sentencia SELECT siguiente determina si las diversas geometrías contenidas en las tablas SAMPLE\_GEOMETRIES1 y SAMPLE\_GEOMETRIES2 forman intersección.

```
SELECT  sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
        sg2.id AS sg2_id, sg2.spatial_type AS sg2_type,
        CASE ST_Intersects(sg1.geometry, sg2.geometry)
            WHEN 0 THEN 'Las geometrías no forman intersección'
            WHEN 1 THEN 'Las geometrías forman intersección'
        END AS intersects
FROM    sample_geometries1 sg1, sample_geometries2 sg2
ORDER BY sg1.id
```

Resultados:

SG1_ID	SG1_TYPE	SG2_ID	SG2_TYPE	INTERSECTS
1	ST_Point	101	ST_Point	Las geometrías forman intersección
1	ST_Point	102	ST_Point	Las geometrías no forman intersección
1	ST_Point	103	ST_Point	Las geometrías no forman intersección
1	ST_Point	110	ST_LineString	Las geometrías no forman intersección
1	ST_Point	120	ST_Polygon	Las geometrías no forman intersección
1	ST_Point	121	ST_Polygon	Las geometrías no forman intersección
10	ST_LineString	101	ST_Point	Las geometrías no forman intersección
10	ST_LineString	102	ST_Point	Las geometrías no forman intersección
10	ST_LineString	103	ST_Point	Las geometrías forman intersección
10	ST_LineString	110	ST_LineString	Las geometrías forman intersección
10	ST_LineString	120	ST_Polygon	Las geometrías no forman intersección
10	ST_LineString	121	ST_Polygon	Las geometrías no forman intersección
20	ST_Polygon	101	ST_Point	Las geometrías forman intersección
20	ST_Polygon	102	ST_Point	Las geometrías forman intersección
20	ST_Polygon	103	ST_Point	Las geometrías no forman intersección
20	ST_Polygon	110	ST_LineString	Las geometrías no forman intersección
20	ST_Polygon	120	ST_Polygon	Las geometrías forman intersección
20	ST_Polygon	121	ST_Polygon	Las geometrías no forman intersección

## ST\_Is3d

ST\_Is3d toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría tiene coordenadas Z. En caso contrario, se devuelve un 0 (cero).

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_Is3D—(—geometría—)—————►►

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría en la que se va a verificar si existen coordenadas Z.

### Tipo de retorno

INTEGER

### Ejemplo

Este ejemplo crea varias geometrías con y sin coordenadas Z y M (medidas). Luego, se utiliza ST\_Is3d para determinar cuál de las geometrías contiene coordenadas Z.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)
```

```
INSERT INTO sample_geoms VALUES
(1, ST_Geometry('point EMPTY',0))
```

```
INSERT INTO sample_geoms VALUES
(2, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))' ,0))
```

```

INSERT INTO sample_geoms VALUES
    (3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
    (4, ST_Geometry('linestring z (10 10 166, 20 10 168)',0))

INSERT INTO sample_geoms VALUES
    (5, ST_Geometry('point zm (10 10 16 30)' ,0))

SELECT id, ST_Is3d(geometry) Is_3D
FROM sample_geoms

```

Resultados:

ID	IS_3D
1	0
2	0
3	0
4	1
5	1

---

## ST\_IsClosed

ST\_IsClosed toma una curva o multicurva como parámetro de entrada y devuelve un 1 si dicha curva o multicurva es cerrada. En caso contrario, se devuelve un 0 (cero).

Una curva es cerrada si el punto inicial y el punto final son iguales. Si la curva tiene coordenadas Z, las coordenadas Z del punto inicial y final deben ser iguales. En caso contrario, los puntos no se consideran iguales y la curva no estará cerrada. Una multicurva es cerrada si cada una de sus curvas es cerrada.

Si la curva o multicurva proporcionada está vacía, se devuelve un 0 (cero). Si es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►► db2gse.ST\_IsClosed(—*curva*—) ◀◀

### Parámetro

**curva** Valor de tipo ST\_Curve o ST\_MultiCurve o uno de sus subtipos que representa la curva o multicurva que se va a verificar.

### Tipo de retorno

INTEGER

### Ejemplos

#### Ejemplo 1

Este ejemplo crea varias cadenas lineales. Las dos últimas cadenas lineales tienen las mismas coordenadas X e Y, pero una cadena lineal contiene coordenadas Z variables que hacen que la cadena lineal no sea cerrada, y las demás cadenas lineales contienen coordenadas M variables (medidas) que no afectan en el hecho de si la cadena lineal es cerrada.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_Linestring)

INSERT INTO sample_lines VALUES
  (1, ST_Linestring('linestring EMPTY',0))

INSERT INTO sample_lines VALUES
  (2, ST_Linestring('linestring(10 10, 20 10, 20 20)' ,0))

INSERT INTO sample_lines VALUES
  (3, ST_Linestring('linestring(10 10, 20 10, 20 20, 10 10)' ,0))

INSERT INTO sample_lines VALUES
  (4, ST_Linestring('linestring m(10 10 1, 20 10 2, 20 20 3,
  10 10 4)' ,0))

INSERT INTO sample_lines VALUES
  (5, ST_Linestring('linestring z(10 10 5, 20 10 6, 20 20 7,
  10 10 8)' ,0))

SELECT id, ST_IsClosed(geometry) Is_Closed
FROM sample_lines
```

Resultados:

ID	IS_CLOSED
1	0
2	0
3	1
4	1
5	0

## Ejemplo 2

Este ejemplo crea dos multilíneas. ST\_IsClosed se utiliza para determinar si las multilíneas están cerradas. La primera multilínea no es cerrada, aunque todas las curvas forman conjuntamente un bucle cerrado completo. Esto es debido a que cada curva individual no es cerrada.

La segunda multilínea es cerrada, pues cada curva individual es cerrada.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER, geometry ST_MultiLinestring)
INSERT INTO sample_mlines VALUES
  (6, ST_MultiLinestring('multilinestring((10 10, 20 10, 20 20),
  (20 20, 30 20, 30 30),
  (30 30, 10 30, 10 10))',0))

INSERT INTO sample_mlines VALUES
  (7, ST_MultiLinestring('multilinestring((10 10, 20 10, 20 20, 10 10 ),
  (30 30, 50 30, 50 50,
  30 30 )',0))

SELECT id, ST_IsClosed(geometry) Is_Closed
FROM sample_mlines
```

Resultados:

ID	IS_CLOSED
6	0
7	1

## ST\_IsEmpty

ST\_IsEmpty toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría está vacía. En caso contrario, se devuelve un 0 (cero).

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►► db2gse.ST\_IsEmpty(—geometría—) ◀◀

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se va a verificar.

### Tipo de retorno

INTEGER

### Ejemplo

El código siguiente crea tres geometrías y luego determina si están vacías.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
  (1, ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
  (2, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))' ,0))

INSERT INTO sample_geoms VALUES
  (3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
  (4, ST_Geometry('linestring z (10 10 166, 20 10 168)',0))

INSERT INTO sample_geoms VALUES
  (5, ST_Geometry('point zm (10 10 16 30)' ,0))
SELECT id, ST_IsEmpty(geometry) Is_Empty
FROM sample_geoms
```

Resultados:

ID	IS_EMPTY
1	1
2	0

3	0
4	0
5	0

---

## ST\_IsMeasured

ST\_IsMeasured toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría tiene coordenadas M (medidas). En caso contrario, se devuelve un 0 (cero).

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►► db2gse.ST\_IsMeasured(*—geometría—*) ◀◀

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría en la que se va a verificar si existen coordenadas M (medidas).

### Tipo de retorno

INTEGER

### Ejemplo

Este ejemplo crea varias geometrías con y sin coordenadas Z y M (medidas). Luego, se utiliza ST\_IsMeasured para determinar cuál de las geometrías contiene medidas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)
```

```
INSERT INTO sample_geoms VALUES
(1, ST_Geometry('point EMPTY',0))
```

```
INSERT INTO sample_geoms VALUES
(2, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))' ,0))
```

```
INSERT INTO sample_geoms VALUES
(3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))
```

```
INSERT INTO sample_geoms VALUES
(4, ST_Geometry('linestring z (10 10 166, 20 10 168)',0))
```

```
INSERT INTO sample_geoms VALUES
(5, ST_Geometry('point zm (10 10 16 30)' ,0))
```

```
SELECT id, ST_IsMeasured(geometry) Is_Measured
FROM sample_geoms
```

Resultados:

ID	IS_MEASURED
1	0
2	0
3	1
4	0
5	1

## ST\_IsRing

ST\_IsRing toma una curva como parámetro de entrada y devuelve un 1 si es un anillo. En caso contrario, se devuelve un 0 (cero). Una curva es un anillo si es simple y cerrada.

Si la curva proporcionada está vacía, se devuelve un 0 (cero). Si es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_IsRing—(—*curva*—)—————►►

### Parámetro

**curva** Valor de tipo ST\_Curve o uno de sus subtipos que representa la curva que se va a verificar.

### Tipo de retorno

INTEGER

### Ejemplos

Este ejemplo crea cuatro cadenas lineales. Luego, se utiliza ST\_IsRing para comprobar si las cadenas lineales son anillos. La última cadena lineal no es considerada un anillo aunque es cerrada, debido a que forma intersección consigo misma.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_Linestring)

INSERT INTO sample_lines VALUES
  (1, ST_Linestring('linestring EMPTY',0))

INSERT INTO sample_lines VALUES
  (2, ST_Linestring('linestring(10 10, 20 10, 20 20)' ,0))

INSERT INTO sample_lines VALUES
  (3, ST_Linestring('linestring(10 10, 20 10, 20 20, 10 10)' ,0))

INSERT INTO sample_lines VALUES
  (4, ST_Linestring('linestring(10 10, 20 10, 10 20, 20 20, 10 10)' ,0))

SELECT id, ST_IsClosed(geometry) Is_Closed, ST_IsRing(geometry) Is_Ring
FROM sample_lines
```

Resultados:



ID	IS_CLOSED	IS_RING
1	1	0
2	0	0
3	1	1
4	1	0

## ST\_IsSimple

ST\_IsSimple toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría es simple. En caso contrario, se devuelve un 0 (cero).

Los puntos, las superficies y multisuperficies son siempre simples. Una curva es simple si no atraviesa el mismo punto dos veces; un multipunto es simple si no contiene dos puntos iguales; y una multicurva es simple si todas sus curvas son simples y las únicas intersecciones se producen en puntos que se encuentran en el límite de las curvas que forman la multicurva.

Si la geometría proporcionada está vacía, se devuelve un 1. Si la geometría es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
db2gse.ST_IsSimple(geometría)
```

### Parámetro

#### **geometría**

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se va a verificar.

### Tipo de retorno

INTEGER

### Ejemplos

Este ejemplo crea varias geometrías y comprueba si son simples. La geometría cuyo ID es 4 no se considera que sea simple porque contiene más de un punto que es el mismo. La geometría cuyo ID es 6 no se considera que sea simple porque la cadena lineal forma intersección consigo misma.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)
```

```
INSERT INTO sample_geoms VALUES
(1, ST_Geometry('point EMPTY' ,0))
```

```
INSERT INTO sample_geoms VALUES
(2, ST_Geometry('point (21 33)' ,0))
```

```
INSERT INTO sample_geoms VALUES
(3, ST_Geometry('multipoint(10 10, 20 20, 30 30)' ,0))
```

```
INSERT INTO sample_geoms VALUES
(4, ST_Geometry('multipoint(10 10, 20 20, 30 30, 20 20)' ,0))
```

```

INSERT INTO sample_geoms VALUES
    (5, ST_Geometry('linestring(60 60, 70 60, 70 70)',0))

INSERT INTO sample_geoms VALUES
    (6, ST_Geometry('linestring(20 20, 30 30, 30 20, 20 30 )',0))

INSERT INTO sample_geoms VALUES
    (7, ST_Geometry('polygon((40 40, 50 40, 50 50, 40 40 ))',0))

SELECT id, ST_IsSimple(geometry) Is_Simple
FROM sample_geoms

```

Resultados:

ID	IS_SIMPLE
1	1
2	1
3	1
4	0
5	1
6	0
7	1

## ST\_IsValid

ST\_IsValid toma una geometría como parámetro de entrada y devuelve un 1 si la geometría es válida. En caso contrario, se devuelve un 0 (cero).

Una geometría es válida sólo si todos los atributos contenidos en el tipo estructurado son coherentes con la representación interna de los datos de la geometría, y si la representación interna no está dañada.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```

▶▶ db2gse.ST_IsValid(—geometría—) ◀◀

```

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos.

### Tipo de retorno

INTEGER

### Ejemplo

Este ejemplo crea varias geometrías y utiliza ST\_IsValid para comprobar su validez. Todas las geometrías son válidas porque las rutinas constructoras, tales como ST\_Geometry, no permiten la creación de geometrías no válidas.

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

```

```

INSERT INTO sample_geoms VALUES

```

```

(1, ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
(2, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))' ,0))

INSERT INTO sample_geoms VALUES
(3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
(4, ST_Geometry('linestring z (10 10 166, 20 10 168)',0))

INSERT INTO sample_geoms VALUES
(5, ST_Geometry('point zm (10 10 16 30)' ,0))

SELECT id, ST_IsValid(geometry) Is_Valid
FROM sample_geoms

```

Resultados:

ID	IS_VALID
1	1
2	1
3	1
4	1
5	1

## ST\_Length

ST\_Length toma como parámetros de entrada una curva o multicurva y, opcionalmente, una unidad de medida, y devuelve la longitud de la curva o multicurva proporcionada, expresada en la unidad de medida proporcionada o en la unidad por omisión.

Si la curva o multicurva proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```

db2gse.ST_Length(curva [, unidad])

```

### Parámetro

**curva** Valor de tipo ST\_Curve o ST\_MultiCurve que representa las curvas para las que se obtiene la longitud.

#### unidad

Valor de tipo VARCHAR(128) que identifica las unidades utilizadas para medir la longitud. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST\_UNITS\_OF\_MEASURE.

Si se omite el parámetro *unidad*, se siguen las normas siguientes para determinar la unidad utilizada para medir la longitud:

- Si *curva* está en un sistema de coordenadas proyectadas o geocéntricas, la unidad lineal asociada a este sistema de coordenadas es el valor por omisión.

- Si *curva* está en un sistema de coordenadas geográficas, pero no está en un sistema de referencia espacial (SRS) geodésico, la unidad angular asociada a este sistema de coordenadas es el valor por omisión.
- Si *curva* está en un SRS geodésico, la unidad por omisión de medida es el metro.

**Restricciones para las conversiones de unidades:** Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La *curva* está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La *curva* está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La *curva* está en un sistema de coordenadas geográficas, pero no en un SRS geodésico y se ha especificado una unidad lineal.
- La *curva* está en un SRS geodésico y se ha especificado una unidad angular.

## Tipo de retorno

DOUBLE

## Ejemplos

### Ejemplo 1

Las siguientes sentencias de SQL crean la tabla SAMPLE\_GEOMETRIES e insertan una línea y una multilínea en la tabla.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_geometries(id SMALLINT, spatial_type varchar(20),
    geometry ST_GEOMETRY)
```

```
INSERT INTO sample_geometries(id, spatial_type, geometry)
VALUES
    (1110, 'ST_LineString', ST_LineString('linestring(50 10, 50 20)', 1)),
    (1111, 'ST_MultiLineString', ST_MultiLineString('multilinestring
        ((33 2, 34 3, 35 6),
         (28 4, 29 5, 31 8, 43 12),
         (39 3, 37 4, 36 7))', 1))
```

### Ejemplo 2

La sentencia SELECT siguiente calcula la longitud de la línea contenida en la tabla SAMPLE\_GEOMTRIES.

```
SELECT id, spatial_type, cast(ST_Length(geometry..ST_ToLineString)
    AS DECIMAL(7, 2)) AS "Line Length"
FROM sample_geometries
WHERE id = 1110
```

Resultados:

ID	SPATIAL_TYPE	Line Length
1110	ST_LineString	10.00

### Ejemplo 3

La sentencia SELECT siguiente calcula la longitud de la multilínea contenida en la tabla SAMPLE\_GEOMTRIES.

```
SELECT id, spatial_type, ST_Length(ST_ToMultiLine(geometry))
      AS multiline_length
FROM sample_geometries
WHERE id = 1111
```

Resultados:

ID	SPATIAL_TYPE	MULTILINE_LENGTH
1111	ST_MultiLineString	+2.76437123387202E+001

## ST\_LineFromText

ST\_LineFromText toma como parámetros de entrada una representación de texto convencional de una cadena lineal y, opcionalmente, un identificador de sistemas de referencia espacial, y devuelve la cadena lineal correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST\_LineString.

### Sintaxis

```
db2gse.ST_LineFromText(wkt, id_srs)
```

### Parámetro

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la cadena lineal resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la cadena lineal resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_LineString

### Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST\_LineFromText para crear e insertar una línea a partir de una representación de texto convencional (WKT) de una línea. Las filas se insertan en la tabla SAMPLE\_LINES con un ID y un valor de línea en el sistema de referencia espacial 1 utilizando una representación WKT.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_lines(id SMALLINT, geometry ST_LineString)
```

```

INSERT INTO sample_lines(id, geometry)
VALUES
  (1110, ST_LineFromText('linestring(850 250, 850 850)', 1) ),
  (1111, ST_LineFromText('linestring empty', 1) )

SELECT id, cast(geometry..ST_AsText AS varchar(75)) AS linestring
FROM sample_lines

```

Resultados:

```

ID      LINESTRING
-----
1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)
1111 LINESTRING EMPTY

```

---

## ST\_LineFromWKB

ST\_LineFromWKB toma como parámetros de entrada una representación binaria convencional de una cadena lineal y, opcionalmente, un identificador de sistemas de referencia espacial, y devuelve la cadena lineal correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST\_LineString.

### Sintaxis

```

▶▶ db2gse.ST_LineFromWKB ( ( wkb [ , id_srs ] ) ) ▶▶

```

### Parámetro

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la cadena lineal resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la cadena lineal resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_LineString

### Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente utiliza la función ST\_LineFromWKB para crear e insertar una línea a partir de una representación binaria convencional. La fila se inserta en la tabla SAMPLE\_LINES con un ID y una línea en el sistema de referencia espacial 1 utilizando una representación WKB.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_lines(id SMALLINT, geometry ST_LineString, wkb BLOB(32k))

INSERT INTO sample_lines(id, geometry)
VALUES
  (1901, ST_LineString('linestring(850 250, 850 850)', 1) ),
  (1902, ST_LineString('linestring(33 2, 34 3, 35 6)', 1) )

UPDATE sample_lines AS temp_correlated
SET wkb = geometry.ST_AsBinary
WHERE id = temp_correlated.id

SELECT id, cast(ST_LineFromWKB(wkb)..ST_AsText AS varchar(90)) AS line
FROM sample_lines
```

Resultados:

```
ID      LINE
-----
1901 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)

1902 LINESTRING ( 33.00000000 2.00000000, 34.00000000 3.00000000,
35.00000000 6.00000000)
```

## ST\_LineString

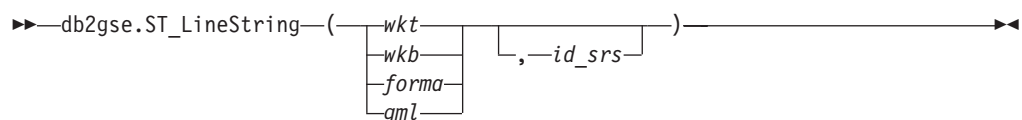
ST\_LineString crea una cadena lineal partir de uno de los datos de entrada siguientes:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma ESRI
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenida la cadena lineal resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma ESRI o la representación GML es nula, se devuelve un valor nulo.

### Sintaxis



### Parámetro

- wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional del polígono resultante.
- wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional del polígono resultante.

**forma** Valor de tipo BLOB(2G) que contiene la representación de forma ESRI del polígono resultante.

**gml** Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), del polígono resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al polígono resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite un error (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_LineString

## Ejemplos

El código siguiente utiliza la función ST\_LineString para crear e insertar una línea a partir de una representación de texto convencional (WKT) de una línea o de una representación binaria convencional.

El ejemplo siguiente inserta una fila en la tabla SAMPLE\_LINES, con un ID y una línea en el sistema de referencia espacial 1, en una representación WKT y GML.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_lines(id SMALLINT, geometry ST_LineString)
```

```
INSERT INTO sample_lines(id, geometry)
```

```
VALUES
```

```
(1110, ST_LineString('linestring(850 250, 850 850)', 1) ),  
(1111, ST_LineString('<gml:LineString srsName=";EPSG:4269";><gml:coord>  
<gml:X>90</gml:X><gml:Y>90</gml:Y>  
</gml:coord><gml:coord><gml:X>100</gml:X>  
<gml:Y>100</gml:Y></gml:coord>  
</gml:LineString>', 1) )
```

```
SELECT id, cast(geometry..ST_AsText AS varchar(75)) AS linestring  
FROM sample_lines
```

Resultados:

```
ID      LINESTRING
```

```
-----  
1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)  
1111 LINESTRING ( 90.00000000 90.00000000, 100.00000000 100.00000000)
```

---

## ST\_LineStringN

ST\_LineStringN toma un multilínea y un índice como parámetros de entrada y devuelve la cadena lineal identificada por el índice. La cadena lineal resultante se representa según el sistema de referencia espacial de la multilínea proporcionada.

Si la multilínea proporcionada tiene un valor nulo o está vacía, o si el índice es menor que 1 o mayor que el número de cadenas lineales, se devuelve un valor nulo.



También se puede invocar esta función como método.

## Sintaxis

► db2gse.ST\_LineStringN(*—multilínea—*, *—índice—*) ◀

## Parámetro

### multilínea

Valor de tipo ST\_MultiLineString que representa la multilínea a partir de la cual se obtiene la cadena lineal identificada por *índice*.

**índice** Valor de tipo INTEGER que identifica la cadena lineal *n* que se obtendrá a partir de *multilínea*.

Si *índice* es menor que 1 o mayor que el número de cadenas lineales de *multilínea*, se devuelve un valor nulo y se emite una condición de aviso (SQLSTATE 01HS0).

## Tipo de retorno

db2gse.ST\_LineString

## Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

La sentencia SELECT selecciona la segunda geometría contenida en una multilínea de la tabla SAMPLE\_MLINES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
CREATE TABLE sample_mlines (id INTEGER,  
    geometry ST_MULTILINESTRING)
```

```
INSERT INTO sample_mlines(id, geometry)  
VALUES  
    (1110, ST_MultiLineString('multilinestring  
        ((33 2, 34 3, 35 6),  
        (28 4, 29 5, 31 8, 43 12),  
        (39 3, 37 4, 36 7))', 1) ),  
    (1111, ST_MLineFromText('multilinestring(  
        (61 2, 64 3, 65 6),  
        (58 4, 59 5, 61 8),  
        (69 3, 67 4, 66 7, 68 9))', 1) )
```

```
SELECT id, cast(ST_LineStringN(geometry, 2)..ST_AsText  
    AS varchar(110)) AS second_linestring  
FROM sample_mlines
```

Resultados:

ID	SECOND_LINSTRING
1110	LINestring ( 28.00000000 4.00000000, 29.00000000 5.00000000, 31.00000000 8.00000000, 43.00000000 12.00000000)
1111	LINestring ( 58.00000000 4.00000000, 59.00000000 5.00000000, 61.00000000 8.00000000)

## ST\_M

ST\_M puede actuar de una de estas maneras:

- Tomar un punto como parámetro de entrada y devolver su coordenada M (medida)
- Tomar como parámetros de entrada un punto y una coordenada M y devolver el propio punto junto con su coordenada M expresada en la medida proporcionada, aunque el punto especificado no tenga ninguna coordenada M.

Si la coordenada M especificada es nula, la coordenada M del punto se elimina.

Si el punto especificado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►► db2gse.ST\_M ( *punto* [ , *coordenada\_m* ] ) ►►

### Parámetros

**punto** Valor de tipo ST\_Point para el cual se obtiene o modifica la coordenada M.

**coordenada\_m**

Valor de tipo DOUBLE que representa la nueva coordenada M del *punto*.

Si *coordenada\_m* es nulo, la coordenada M se elimina de *punto*.

### Tipos de retorno

- DOUBLE, si no se especifica *coordenada\_m*
- db2gse.ST\_Point, si se especifica *coordenada\_m*

### Ejemplos

#### Ejemplo 1

En este ejemplo, se muestra el uso de la función ST\_M. Se crean tres puntos y luego se insertan en la tabla SAMPLE\_POINTS. Todos ellos están en el sistema de referencia espacial cuyo ID es 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
```

```
INSERT INTO sample_points
VALUES (1, ST_Point (2, 3, 32, 5, 1))
```

```
INSERT INTO sample_points
VALUES (2, ST_Point (4, 5, 20, 4, 1))
```

```
INSERT INTO sample_points
VALUES (3, ST_Point (3, 8, 23, 7, 1))
```

#### Ejemplo 2

Este ejemplo determina la coordenada M de los puntos contenidos en la tabla SAMPLE\_POINTS.

```
SELECT id, ST_M (geometry) M_COORD
FROM sample_points
```

Resultados:

ID	M_COORD
1	+5.000000000000000E+000
2	+4.000000000000000E+000
3	+7.000000000000000E+000

### Ejemplo 3

Este ejemplo obtiene uno de los puntos junto con su coordenada M establecida en el valor 40.

```
SELECT id, CAST (ST_AsText (ST_M (geometry, 40) )
AS VARCHAR(60) ) M_COORD_40
FROM sample_points
WHERE id=3
```

Resultados:

ID	M_COORD_40
3	POINT ZM (3.00000000 8.00000000 23.00000000 40.00000000)

---

## ST\_MaxM

ST\_MaxM toma una geometría como parámetro de entrada y devuelve su coordenada M máxima.

Si la geometría proporcionada es un valor nulo o está vacía, o si no tiene coordenadas M, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
db2gse.ST_MaxM(geometría)
```

### Parámetro

#### **geometría**

Valor de tipo ST\_Geometry o uno de sus subtipos para el que se devuelve la coordenada M máxima.

### Tipo de retorno

DOUBLE

### Ejemplos

#### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_MaxM. Se crean tres polígonos y luego se insertan en la tabla SAMPLE\_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                110 140 22 3,
                                120 130 26 4,
                                110 120 20 3))', 0) )
```

```
INSERT INTO sample_polys
VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                0 4 35 9,
                                5 4 32 12,
                                5 0 31 5,
                                0 0 40 7))', 0) )
```

```
INSERT INTO sample_polys
VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                8 4 10 12,
                                9 4 12 11,
                                12 13 10 16))', 0) )
```

### Ejemplo 2

Este ejemplo determina la coordenada M máxima de cada polígono contenido en SAMPLE\_POLYS.

```
SELECT id, CAST ( ST_MaxM(geometry) AS INTEGER) MAX_M
FROM sample_polys
```

Resultados:

ID	MAX_M
1	4
2	12
3	16

### Ejemplo 3

Este ejemplo determina la coordenada M máxima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxM(geometry) ) AS INTEGER) OVERALL_MAX_M
FROM sample_polys
```

Resultados:

OVERALL_MAX_M
16

---

## ST\_MaxX

ST\_MaxX toma una geometría como parámetro de entrada y devuelve su coordenada X máxima.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos para el que se devuelve la coordenada X máxima.

## Tipo de retorno

DOUBLE

## Ejemplos

### Ejemplo 1

Este ejemplo ilustra el uso de la función ST\_MaxX. Se crean tres polígonos y luego se insertan en la tabla SAMPLE\_POLYS. El tercer ejemplo muestra cómo utilizar todas las funciones que obtienen las coordenadas máxima y mínima para evaluar el rango espacial de las geometrías contenidas en una determinada columna espacial.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                110 140 22 3,
                                120 130 26 4,
                                110 120 20 3))', 0) )
```

```
INSERT INTO sample_polys
VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                0 4 35 9,
                                5 4 32 12,
                                5 0 31 5,
                                0 0 40 7))', 0) )
```

```
INSERT INTO sample_polys
VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                8 4 10 12,
                                9 4 12 11,
                                12 13 10 16))', 0) )
```

### Ejemplo 2

Este ejemplo determina la coordenada X máxima de cada polígono contenido en SAMPLE\_POLYS.

```
SELECT id, CAST ( ST_MaxX(geometry) AS INTEGER) MAX_X_COORD
FROM sample_polys
```

Resultados:

ID	MAX_X_COORD
1	120
2	5
3	12

### Ejemplo 3

Este ejemplo determina la coordenada X máxima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxX(geometry) ) AS INTEGER) OVERALL_MAX_X
FROM sample_polys
```

Resultados:

```
OVERALL_MAX_X
-----
                120
```

#### Ejemplo 4

Este ejemplo determina el rango espacial (diferencia entre el valor mínimo total y el valor máximo total) de todos los polígonos contenidos en la tabla SAMPLE\_POLYS. Este cálculo se suele utilizar para comparar el rango espacial actual de las geometrías con el rango espacial del sistema de referencia espacial de los datos, para determinar si existe espacio para que el crecimiento de los datos.

```
SELECT CAST ( MIN (ST_MinX (geometry)) AS INTEGER) MIN_X,
       CAST ( MIN (ST_MinY (geometry)) AS INTEGER) MIN_Y,
       CAST ( MIN (ST_MinZ (geometry)) AS INTEGER) MIN_Z,
       CAST ( MIN (ST_MinM (geometry)) AS INTEGER) MIN_M,
       CAST ( MAX (ST_MaxX (geometry)) AS INTEGER) MAX_X,
       CAST ( MAX (ST_MaxY (geometry)) AS INTEGER) MAX_Y,
       CAST ( MAX (ST_MaxZ (geometry)) AS INTEGER) MAX_Z,
       CAST ( MAX (ST_MaxM(geometry)) AS INTEGER) MAX_M,
FROM sample_polys
```

Resultados:

```
MIN_X    MIN_Y    MIN_Z    MIN_M    MAX_X    MAX_Y    MAX_Z    MAX_M
-----
          0         0         10         3        120        140         40         16
```

---

## ST\_MaxY

ST\_MaxY toma una geometría como parámetro de entrada y devuelve su coordenada Y máxima.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
►►—db2gse.ST_MaxY—(—geometría—)—————►►
```

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos para el que se devuelve la coordenada Y máxima.

### Tipo de retorno

DOUBLE

## Ejemplos

### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_MaxY. Se crean tres polígonos y luego se insertan en la tabla SAMPLE\_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                110 140 22 3,
                                120 130 26 4,
                                110 120 20 3))', 0) )
```

```
INSERT INTO sample_polys
VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                0 4 35 9,
                                5 4 32 12,
                                5 0 31 5,
                                0 0 40 7))', 0) )
```

```
INSERT INTO sample_polys
VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                8 4 10 12,
                                9 4 12 11,
                                12 13 10 16))', 0) )
```

### Ejemplo 2

Este ejemplo determina la coordenada Y máxima de cada polígono contenido en SAMPLE\_POLYS.

```
SELECT id, CAST ( ST_MaxY(geometry) AS INTEGER) MAX_Y
FROM sample_polys
```

Resultados:

ID	MAX_Y
1	140
2	4
3	13

### Ejemplo 3

Este ejemplo determina la coordenada Y máxima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxY(geometry) ) AS INTEGER) OVERALL_MAX_Y
FROM sample_polys
```

Resultados:

OVERALL_MAX_Y
140

---

## ST\_MaxZ

ST\_MaxZ toma una geometría como parámetro de entrada y devuelve su coordenada Z máxima.

Si la geometría proporcionada es un valor nulo o está vacía, o si no tiene coordenadas Z, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►►—db2gse.ST\_MaxZ—(—geometría—)—————►►

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos para el que se devuelve la coordenada Z máxima.

## Tipo de retorno

DOUBLE

## Ejemplos

### Ejemplo 1

En este ejemplo, se muestra el uso de la función ST\_MaxZ. Se crean tres polígonos y luego se insertan en la tabla SAMPLE\_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                               110 140 22 3,
                               120 130 26 4,
                               110 120 20 3))', 0) )
```

```
INSERT INTO sample_polys
VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                               0 4 35 9,
                               5 4 32 12,
                               5 0 31 5,
                               0 0 40 7))', 0) )
```

```
INSERT INTO sample_polys
VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                               8 4 10 12,
                               9 4 12 11,
                               12 13 10 16))', 0) )
```

### Ejemplo 2

Este ejemplo determina la coordenada Z máxima de cada polígono contenido en SAMPLE\_POLYS.

```
SELECT id, CAST ( ST_MaxZ(geometry) AS INTEGER) MAX_Z
FROM sample_polys
```

Resultados:



ID	MAX_Z
1	26
2	40
3	12

### Ejemplo 3

Este ejemplo determina la coordenada Z máxima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxZ(geometry) ) AS INTEGER) OVERALL_MAX_Z
FROM sample_polys
```

Resultados:

```
OVERALL_MAX_Z
-----
40
```

---

## ST\_MBR

ST\_MBR toma una geometría como parámetro de entrada y devuelve el rectángulo delimitador mínimo de la misma.

Si dicha geometría es un punto, se devuelve el propio punto. Si la geometría es una cadena lineal horizontal o vertical y el sistema de referencia espacial no es geodésico, se devuelve la propia cadena lineal horizontal o vertical. En otro caso, se devuelve el rectángulo delimitador mínimo de la geometría en forma de polígono. Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►► db2gse.ST\_MBR(—*geometría*—) ◀◀

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la que se obtiene el rectángulo delimitador mínimo.

### Tipo de retorno

db2gse.ST\_Geometry

### Ejemplo

Este ejemplo muestra cómo utilizar la función ST\_MBR para obtener el rectángulo delimitador mínimo de un polígono. Debido a que la geometría especificada es un polígono, el rectángulo delimitador mínimo se devuelve en forma de polígono.

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
VALUES (1, ST_Polygon ('polygon (( 5 5, 7 7, 5 9, 7 9, 9 11, 13 9,
                               15 9, 13 7, 15 5, 9 6, 5 5))', 0) )

INSERT INTO sample_polys
VALUES (2, ST_Polygon ('polygon (( 20 30, 25 35, 30 30, 20 30))', 0) )

SELECT id, CAST (ST_AsText ( ST_MBR(geometry)) AS VARCHAR(150) ) MBR
FROM sample_polys

```

Resultados:

ID	MBR
1	POLYGON (( 5.00000000 5.00000000, 15.00000000 5.00000000, 15.00000000 11.00000000, 5.00000000 11.00000000, 5.00000000 5.00000000))
2	POLYGON (( 20.00000000 30.00000000, 30.00000000 30.00000000, 30.00000000 35.00000000, 20.00000000 35.00000000, 20.00000000 30.00000000 ))

## ST\_MBRIntersects

ST\_MBRIntersects toma dos geometrías como parámetros de entrada y devuelve un 1 si los rectángulos delimitadores mínimos de las dos geometrías forman intersección. En caso contrario, se devuelve un 0 (cero). El rectángulo delimitador mínimo de un punto y de una cadena lineal horizontal o vertical es la propia geometría.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las dos geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

### Sintaxis

►►—db2gse.ST\_MBRIntersects—(—*geometría1*—,—*geometría2*—)—————►

### Parámetros

#### **geometría1**

Un valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría cuyo rectángulo delimitador mínimo que se debe comprobar si forma intersección con el rectángulo delimitador mínimo de *geometría2*.

#### **geometría2**

Un valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría cuyo rectángulo delimitador mínimo que se debe comprobar para ver si forma intersección con el rectángulo delimitador mínimo de *geometría1*.

### Tipo de retorno

INTEGER

## Ejemplos

### Ejemplo 1

Este ejemplo muestra el uso de ST\_MBRIntersects para obtener una aproximación de si dos polígonos inconexos están próximos entre sí observando si sus rectángulos delimitadores mínimos forman intersección. El primer ejemplo utiliza la expresión CASE de SQL. El segundo ejemplo utiliza una sentencia SELECT individual para encontrar los polígonos que forman intersección con el rectángulo delimitador mínimo del polígono cuyo ID es 2.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
VALUES (1, ST_Polygon ('polygon (( 0 0, 30 0, 40 30, 40 35,
                               5 35, 5 10, 20 10, 20 5, 0 0 ))', 0) )

INSERT INTO sample_polys
VALUES (2, ST_Polygon ('polygon (( 15 15, 15 20, 60 20, 60 15,
                               15 15 ))', 0) )

INSERT INTO sample_polys
VALUES (3, ST_Polygon ('polygon (( 115 15, 115 20, 160 20, 160 15,
                               115 15 ))', 0) )
```

### Ejemplo 2

La sentencia SELECT siguiente utiliza una expresión CASE para encontrar los ID de los polígonos que tienen rectángulos delimitadores mínimos que forman intersección.

```
SELECT a.id, b.id,
       CASE ST_MBRIntersects (a.geometry, b.geometry)
         WHEN 0 THEN 'MBRs do not intersect'
         WHEN 1 THEN 'MBRs intersect'
       END AS MBR_INTERSECTS
FROM sample_polys a, sample_polys b
WHERE a.id <= b.id
```

Resultados:

ID	ID	MBR_INTERSECTS
1	1	1 MBRs intersect
1	2	2 MBRs intersect
2	2	2 MBRs intersect
1	3	3 MBRs do not intersect
2	3	3 MBRs do not intersect
3	3	3 MBRs intersect

### Ejemplo 3

La sentencia SELECT siguiente determina si los rectángulos delimitadores mínimos de las geometrías forman intersección con el del polígono cuyo ID es 2.

```
SELECT a.id, b.id, ST_MBRIntersects (a.geometry, b.geometry) MBR_INTERSECTS
FROM sample_polys a, sample_polys b
WHERE a.id = 2
```

Resultados

ID	ID	MBR_INTERSECTS
2	1	1
2	2	1
2	3	0

## ST\_MeasureBetween o ST\_LocateBetween

ST\_MeasureBetween o ST\_LocateBetween toma como parámetros de entrada una geometría y dos coordenadas M (medidas) y devuelve la parte de la geometría proporcionada que representa el conjunto de puntos desconectados situados entre las dos coordenadas M.

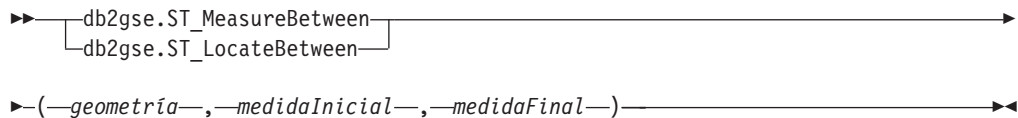
En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si la geometría proporcionada es una superficie o multisuperficie, ST\_MeasureBetween o ST\_LocateBetween se aplica a los anillos exterior e interior de la geometría. Si ninguna de las partes de la geometría proporcionada está dentro del intervalo definido por las coordenadas M proporcionadas, el resultado es una geometría vacía. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

Si la geometría resultante no está vacía, se devuelve un tipo multipunto o multilínea.

Ambas funciones también se pueden invocar como métodos.

### Sintaxis



### Parámetros

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría donde se deben localizar las partes cuyos valores de medida están comprendidos entre *medidaInicial* y *medidaFinal*.

#### medidaInicial

Valor de tipo DOUBLE que representa el límite inferior del intervalo de medidas. Si este valor es nulo, no se aplica ningún límite inferior.

#### medidaFinal

Valor de tipo DOUBLE que representa el límite superior del intervalo de medidas. Si este valor es nulo, no se aplica ningún límite superior.

### Tipo de retorno

db2gse.ST\_Geometry

## Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

La coordenada M (medida) de una geometría está definida por el usuario. Es muy versátil porque puede representar cualquier entidad que desee medir; por ejemplo, una distancia de carretera, o medidas de temperatura, presión o pH.

Este ejemplo muestra el uso de la coordenada M para registrar datos recogidos de mediciones de pH. Un investigador obtiene la medida del pH del suelo en lugares determinados de una carretera. De acuerdo con sus técnicas habituales de trabajo, anota los valores necesarios en cada lugar donde toma una muestra del suelo: las coordenadas X e Y de ese lugar y el pH medido.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_LineString)

INSERT INTO sample_lines
VALUES (1, ST_LineString ('linestring m (2 2 3, 3 5 3,
                          3 3 6, 4 4 6,
                          5 5 6, 6 6 8)', 1 ) )
```

Para encontrar el tramo donde la acidez del suelo oscila entre 4 y 6, el investigador utilizaría esta sentencia SELECT:

```
SELECT id, CAST( ST_AsText( ST_MeasureBetween( 4, 6 )
AS VARCHAR(150) ) MEAS_BETWEEN_4_AND_6
FROM sample_lines
```

Resultados:

ID	MEAS_BETWEEN_4_AND_6
1	LINESTRING M (3.00000000 4.33333300 4.00000000, 3.00000000 3.00000000 6.00000000, 4.00000000 4.00000000 6.00000000, 5.00000000 5.00000000 6.00000000)

---

## ST\_MidPoint

ST\_MidPoint toma una curva como parámetro de entrada y devuelve un punto en la curva que es equidistante de los dos puntos finales de la curva, medido a lo largo de la curva. El punto resultante se representa según el sistema de referencia espacial de la curva proporcionada.

Si la curva proporcionada está vacía, se devuelve un punto vacío. Si la curva proporcionada es un valor nulo, se devuelve un valor nulo.

Si la curva contiene coordenadas Z o coordenadas M (medidas), el punto medio se determina basándose solamente en los valores de las coordenadas Y e Y de la curva. La coordenada Z y la medida del punto obtenido se interpolan.

También se puede invocar esta función como método.

### Sintaxis

```
►►—db2gse.ST_MidPoint—(—curva—)—————►►
```

## Parámetro

**curva** Valor de tipo ST\_Curve o uno de sus subtipos que representa la curva para la que se obtiene el punto medio.

## Tipo de retorno

db2gse.ST\_Point

## Ejemplo

Este ejemplo muestra el uso de ST\_MidPoint para obtener el punto medio de curvas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_LineString)

INSERT INTO sample_lines (id, geometry)
VALUES (1, ST_LineString ('linestring (0 0, 0 10, 0 20, 0 30, 0 40)', 1 ) )

INSERT INTO sample_lines (id, geometry)
VALUES (2, ST_LineString ('linestring (2 2, 3 5, 3 3, 4 4, 5 5, 6 6)', 1 ) )

INSERT INTO sample_lines (id, geometry)
VALUES (3, ST_LineString ('linestring (0 10, 0 0, 10 0, 10 10)', 1 ) )

INSERT INTO sample_lines (id, geometry)
VALUES (4, ST_LineString ('linestring (0 20, 5 20, 10 20, 15 20)', 1 ) )

SELECT id, CAST( ST_AsText( ST_MidPoint(geometry) ) AS VARCHAR(60) ) MID_POINT
FROM sample_lines
```

Resultados:

ID	MID_POINT
1	POINT ( 0.00000000 20.00000000)
2	POINT ( 3.00000000 3.45981800)
3	POINT ( 5.00000000 0.00000000)
4	POINT ( 7.50000000 20.00000000)

---

## ST\_MinM

ST\_MinM toma una geometría como parámetro de entrada y devuelve su coordenada M mínima.

Si la geometría proporcionada es un valor nulo o está vacía, o si no tiene coordenadas M, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►► db2gse.ST\_MinM(—geometría—) ◀◀

## Parámetro

**geometría**

Valor de tipo ST\_Geometry o uno de sus subtipos para el que se devuelve la coordenada M mínima.

## Tipo de retorno

DOUBLE

## Ejemplos

### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_MinM. Se crean tres polígonos y luego se insertan en la tabla SAMPLE\_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                110 140 22 3,
                                120 130 26 4,
                                110 120 20 3))', 0) )
```

```
INSERT INTO sample_polys
VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                0 4 35 9,
                                5 4 32 12,
                                5 0 31 5,
                                0 0 40 7))', 0) )
```

```
INSERT INTO sample_polys
VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                8 4 10 12,
                                9 4 12 11,
                                12 13 10 16))', 0) )
```

### Ejemplo 2

Este ejemplo determina la coordenada M mínima de cada polígono contenido en SAMPLE\_POLYS.

```
SELECT id, CAST ( ST_MinM(geometry) AS INTEGER) MIN_M
FROM sample_polys
```

Resultados:

ID	MIN_M
1	3
2	5
3	11

### Ejemplo 3

Este ejemplo determina la coordenada M mínima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinM(geometry) ) AS INTEGER) OVERALL_MIN_M
FROM sample_polys
```

Resultados:

OVERALL_MIN_M
3

---

## ST\_MinX

ST\_MinX toma una geometría como parámetro de entrada y devuelve la coordenada X mínima.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

► db2gse.ST\_MinX(*—geometría—*) ◀

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos para el que se devuelve la coordenada X mínima.

### Tipo de retorno

DOUBLE

### Ejemplos

#### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_MinX. Se crean tres polígonos y luego se insertan en la tabla SAMPLE\_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                               110 140 22 3,
                               120 130 26 4,
                               110 120 20 3))', 0) )
```

```
INSERT INTO sample_polys
VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                               0 4 35 9,
                               5 4 32 12,
                               5 0 31 5,
                               0 0 40 7))', 0) )
```

```
INSERT INTO sample_polys
VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                               8 4 10 12,
                               9 4 12 11,
                               12 13 10 16))', 0) )
```

#### Ejemplo 2

Este ejemplo determina la coordenada X mínima de cada polígono contenido en SAMPLE\_POLYS.

```
SELECT id, CAST ( ST_MinX(geometry) AS INTEGER) MIN_X
FROM sample_polys
```



Resultados:

ID	MIN_X
1	110
2	0
3	8

### Ejemplo 3

Este ejemplo determina la coordenada X mínima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinX(geometry) ) AS INTEGER) OVERALL_MIN_X
FROM sample_polys
```

Resultados:

OVERALL_MIN_X
0

---

## ST\_MinY

ST\_MinY toma una geometría como parámetro de entrada y devuelve su coordenada Y mínima.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
db2gse.ST_MinY(geometría)
```

### Parámetro

#### *geometría*

Valor de tipo ST\_Geometry o uno de sus subtipos para el que se devuelve la coordenada Y mínima.

### Tipo de retorno

DOUBLE

### Ejemplos

#### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_MinY. Se crean tres polígonos y luego se insertan en la tabla SAMPLE\_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                               110 140 22 3,
                               120 130 26 4,
                               110 120 20 3))', 0) )
```

```

INSERT INTO sample_polys
VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                0 4 35 9,
                                5 4 32 12,
                                5 0 31 5,
                                0 0 40 7))', 0) )

```

```

INSERT INTO sample_polys
VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                8 4 10 12,
                                9 4 12 11,
                                12 13 10 16))', 0) )

```

### Ejemplo 2

Este ejemplo determina la coordenada Y mínima de cada polígono contenido en SAMPLE\_POLYS.

```

SELECT id, CAST ( ST_MinY(geometry) AS INTEGER) MIN_Y
FROM sample_polys

```

Resultados:

ID	MIN_Y
1	120
2	0
3	4

### Ejemplo 3

Este ejemplo determina la coordenada Y mínima existente para todos los polígonos de la columna GEOMETRY.

```

SELECT CAST ( MIN ( ST_MinY(geometry) ) AS INTEGER) OVERALL_MIN_Y
FROM sample_polys

```

Resultados:

OVERALL_MIN_Y
0

---

## ST\_MinZ

ST\_MinZ toma una geometría como parámetro de entrada y devuelve su coordenada Z mínima.

Si la geometría proporcionada es un valor nulo o está vacía, o si no tiene coordenadas Z, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_MinZ—(—geometría—)—————◄◄

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos para el que se devuelve la coordenada Z mínima.

## Tipo de retorno

DOUBLE

## Ejemplos

### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_MinZ. Se crean tres polígonos y luego se insertan en la tabla SAMPLE\_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                110 140 22 3,
                                120 130 26 4,
                                110 120 20 3))', 0) )
```

```
INSERT INTO sample_polys
VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                0 4 35 9,
                                5 4 32 12,
                                5 0 31 5,
                                0 0 40 7))', 0) )
```

```
INSERT INTO sample_polys
VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                8 4 10 12,
                                9 4 12 11,
                                12 13 10 16))', 0) )
```

### Ejemplo 2

Este ejemplo determina la coordenada Z mínima de cada polígono contenido en SAMPLE\_POLYS.

```
SELECT id, CAST ( ST_MinZ(geometry) AS INTEGER) MIN_Z
FROM sample_polys
```

Resultados:

ID	MIN_Z
1	20
2	31
3	10

### Ejemplo 3

Este ejemplo determina la coordenada Z mínima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinZ(geometry) ) AS INTEGER) OVERALL_MIN_Z
FROM sample_polys
```

Resultados:

---

## ST\_MLineFromText

ST\_MLineFromText toma como parámetros de entrada una representación de texto convencional de una multilínea y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la multilínea correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_MultiLineString. Es recomendable debido a su flexibilidad: ST\_MultiLineString acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

### Sintaxis

```
db2gse.ST_MLineFromText( (wkt, id_srs) )
```

### Parámetros

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la multilínea resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la multilínea resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_MultiLineString

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_MLineFromText para crear e insertar una multilínea a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es una multilínea representada en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son:

- Línea 1: (33, 2) (34, 3) (35, 6)
- Línea 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Línea 3: (39, 3) (37, 4) (36, 7)

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER, geometry ST_MultiLineString)

INSERT INTO sample_mlines
VALUES (1110, ST_MLineFromText ('multilinestring ( (33 2, 34 3, 35 6),
                                                (28 4, 29 5, 31 8, 43 12),
                                                (39 3, 37 4, 36 7) )', 1) )

```

La sentencia SELECT siguiente devuelve la multilínea que se registró en la tabla:

```

SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTI_LINE_STRING
FROM sample_mlines
WHERE id = 1110

```

Resultados:

ID	MULTI_LINE_STRING
1110	MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000, 35.00000000 6.00000000), ( 28.00000000 4.00000000, 29.00000000 5.00000000, 31.00000000 8.00000000, 43.00000000 12.00000000), ( 39.00000000 3.00000000, 37.00000000 4.00000000, 36.00000000 7.00000000 ))

## ST\_MLineFromWKB

ST\_MLineFromWKB toma como parámetros de entrada una representación binaria convencional de una multilínea y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la multilínea correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_MultiLineString. Es recomendable debido a su flexibilidad: ST\_MultiLineString acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

### Sintaxis

```

db2gse.ST_MLineFromWKB( ( wkb [, id_srs] ) )

```

### Parámetros

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la multilínea resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la multilínea resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_MultiLineString

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_MLineFromWKB para crear e insertar una multilínea a partir de su representación binaria convencional. La geometría es una multilínea en el sistema de referencia espacial 1. En este ejemplo, la multilínea se guarda con el ID = 10 en la columna GEOMETRY de la tabla SAMPLE\_MLINES, y luego la columna WKB se actualiza con su representación binaria convencional (utilizando la función ST\_AsBinary). Finalmente, se utiliza la función ST\_MLineFromWKB para obtener la multilínea a partir de la columna WKB. Las coordenadas X e Y de esta geometría son:

- Línea 1: (61, 2) (64, 3) (65, 6)
- Línea 2: (58, 4) (59, 5) (61, 8)
- Línea 3: (69, 3) (67, 4) (66, 7) (68, 9)

La tabla SAMPLE\_MLINES tiene una columna GEOMETRY, donde se guarda la multilínea, y una columna WKB, donde se guarda la representación binaria convencional de la multilínea.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER, geometry ST_MultiLineString,
                             wkb BLOB(32K))
```

```
INSERT INTO sample_mlines
VALUES (10, ST_MultiLineString ('multilinestring
    ( (61 2, 64 3, 65 6),
      (58 4, 59 5, 61 8),
      (69 3, 67 4, 66 7, 68 9) )', 1) )
```

```
UPDATE sample_mlines AS temporary_correlated
SET wkb = ST_AsBinary( geometry )
WHERE id = temporary_correlated.id
```

En la sentencia SELECT siguiente, se utiliza la función ST\_MLineFromWKB para obtener la multilínea a partir de la columna WKB.

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb) )
                AS VARCHAR(280) ) MULTI_LINE_STRING
FROM sample_mlines
WHERE id = 10
```

Resultados:

```
ID          MULTI_LINE_STRING
-----
10 MULTILINESTRING (( 61.00000000 2.00000000, 64.00000000 3.00000000,
                      65.00000000 6.00000000),
                    ( 58.00000000 4.00000000, 59.00000000 5.00000000,
                      61.00000000 8.00000000),
                    ( 69.00000000 3.00000000, 67.00000000 4.00000000,
                      66.00000000 7.00000000, 68.00000000 9.00000000 ))
```

---

## ST\_MPointFromText

ST\_MPointFromText toma como parámetros de entrada una representación de texto convencional de un multipunto y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el multipunto correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_MultiPoint. Es recomendable debido a su flexibilidad: ST\_MultiPoint acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

### Sintaxis

```
db2gse.ST_MPointFromText(wkt [, id_srs])
```

### Parámetros

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional del multipunto resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipunto resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_MultiPoint

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_MPointFromText para crear e insertar un multipunto a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un multipunto representada en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son: (1, 2) (4, 3) (5, 6).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpoints (id INTEGER, geometry ST_MultiPoint)
```

```
INSERT INTO sample_mpoints
VALUES (1110, ST_MPointFromText ('multipoint (1 2, 4 3, 5 6)'), 1)
```

La sentencia SELECT siguiente devuelve el multipunto que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTIPOINT
FROM sample_mpoints
WHERE id = 1110
```

Resultados:

```
ID          MULTIPOINT
-----
1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000 3.00000000,
                5.00000000 6.00000000)
```

---

## ST\_MPointFromWKB

ST\_MPointFromWKB toma como parámetros de entrada una representación binaria convencional de un multipunto y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el multipunto correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_MultiPoint. Es recomendable debido a su flexibilidad: ST\_MultiPoint acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

### Sintaxis

```
db2gse.ST_MPointFromWKB(wkb, id_srs)
```

### Parámetros

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional del multipunto resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipunto resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_MultiPoint

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_MPointFromWKB para crear e insertar un multipunto a partir de su representación binaria convencional. La geometría es un multipunto en el sistema de referencia espacial 1. En este ejemplo, el multipunto se guarda con el ID = 10 en la columna GEOMETRY de la tabla SAMPLE\_MPOINTS, y luego la columna WKB se actualiza con su representación binaria convencional (utilizando la función ST\_AsBinary). Finalmente, se utiliza la función ST\_MPointFromWKB para obtener el multipunto a partir de la columna WKB. Las coordenadas X e Y de esta geometría son: (44, 14) (35, 16) (24, 13).



La tabla SAMPLE\_MPOINTS tiene una columna GEOMETRY, donde se guarda el multipunto, y una columna WKB, donde se guarda la representación binaria convencional del multipunto.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpoints (id INTEGER, geometry ST_MultiPoint,
                             wkb BLOB(32K))

INSERT INTO sample_mpoints
VALUES (10, ST_MultiPoint ('multipoint ( 4 14, 35 16, 24 13)', 1))

UPDATE sample_mpoints AS temporary_correlated
SET wkb = ST_AsBinary( geometry )
WHERE id = temporary_correlated.id
```

En la sentencia SELECT siguiente, se utiliza la función ST\_MPointFromWKB para obtener el multipunto a partir de la columna WKB.

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb)) AS VARCHAR(100)) MULTIPOINT
FROM sample_mpoints
WHERE id = 10
```

Resultados:

ID	MULTIPOINT
10	MULTIPOINT (44.00000000 14.00000000, 35.00000000 16.00000000 24.00000000 13.00000000)

## ST\_MPolyFromText

ST\_MPolyFromText toma como parámetros de entrada una representación de texto convencional de un multipolígono y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el multipolígono correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_MultiPolygon. Es recomendable debido a su flexibilidad: ST\_MultiPolygon acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

### Sintaxis

```
►► db2gse.ST_MPolyFromText ( ( wkt ) [ , id_srs ] )
```

### Parámetros

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional del multipolígono resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipolígono resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_MultiPolygon

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_MPolyFromText para crear e insertar un multipolígono a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un multipolígono representado en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son:

- Polígono 1: (3, 3) (4, 6) (5, 3) (3, 3)
- Polígono 2: (8, 24) (9, 25) (1, 28) (8, 24)
- Polígono 3: (13, 33) (7, 36) (1, 40) (10, 43) (13, 33)

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)
```

```
INSERT INTO sample_mpolys
VALUES (1110,
       ST_MPolyFromText ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                           (8 24, 9 25, 1 28, 8 24),
                                           (13 33, 7 36, 1 40, 10 43 13 33) ))', 1) )
```

La sentencia SELECT siguiente devuelve la multipolígono que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
FROM sample_mpolys
WHERE id = 1110
```

Resultados:

```
ID          MULTI_POLYGON
-----
1110 MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
1.00000000 40.00000000, 7.00000000 36.00000000,
13.00000000 33.00000000)),
(( 8.00000000 24.00000000, 9.00000000 25.00000000,
1.00000000 28.00000000, 8.00000000 24.00000000)),
( 3.00000000 3.00000000, 5.00000000 3.00000000,
4.00000000 6.00000000, 3.00000000 3.00000000)))
```

---

## ST\_MPolyFromWKB

ST\_MPolyFromWKB toma como parámetros de entrada una representación binaria convencional de un multipolígono y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el multipolígono correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_MultiPolygon. Es recomendable debido a su flexibilidad: ST\_MultiPolygon acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

## Sintaxis

```
db2gse.ST_MPolyFromWKB ( wkb , id_srs )
```

## Parámetros

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional del multipolígono resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipolígono resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_MultiPolygon

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_MPolyFromWKB para crear un multipolígono a partir de su representación binaria convencional. La geometría es un multipolígono en el sistema de referencia espacial 1. En este ejemplo, el multipolígono se guarda con el ID = 10 en la columna GEOMETRY de la tabla SAMPLE\_MPOLYS, y luego la columna WKB se actualiza con su representación binaria convencional (utilizando la función ST\_AsBinary). Finalmente, se utiliza la función ST\_MPolyFromWKB para obtener el multipolígono a partir de la columna WKB. Las coordenadas X e Y de esta geometría son:

- Polígono 1: (1, 72) (4, 79) (5, 76) (1, 72)
- Polígono 2: (10, 20) (10, 40) (30, 41) (10, 20)
- Polígono 3: (9, 43) (7, 44) (6, 47) (9, 43)

La tabla SAMPLE\_MPOLYS tiene una columna GEOMETRY, donde se guarda el multipolígono, y una columna WKB, donde se guarda la representación binaria convencional del multipolígono.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER,
    geometry ST_MultiPolygon, wkb BLOB(32K))

INSERT INTO sample_mpolys
VALUES (10, ST_MultiPolygon ('multipolygon
    (( (1 72, 4 79, 5 76, 1 72),
    (10 20, 10 40, 30 41, 10 20),
    (9 43, 7 44, 6 47, 9 43) ))', 1))

UPDATE sample_mpolys AS temporary_correlated
SET wkb = ST_AsBinary( geometry )
WHERE id = temporary_correlated.id
```

En la sentencia SELECT siguiente, se utiliza la función ST\_MPolyFromWKB para obtener el multipolígono a partir de la columna WKB.

```
SELECT id, CAST( ST_AsText( ST_MPolyFromWKB (wkb) )
AS VARCHAR(320) ) MULTIPOLYGON
FROM sample_mpolys
WHERE id = 10
```

Resultados:

```
ID          MULTIPOLYGON
-----
10 MULTIPOLYGON ((( 10.00000000 20.00000000, 30.00000000
41.00000000, 10.00000000 40.00000000, 10.00000000
20.00000000)),
( 1.00000000 72.00000000, 5.00000000
76.00000000, 4.00000000 79.00000000, 1.00000000
72,00000000)),
( 9.00000000 43.00000000, 6.00000000
47.00000000, 7.00000000 44.00000000, 9.00000000
43.00000000 )))
```

## ST\_MultiLineString

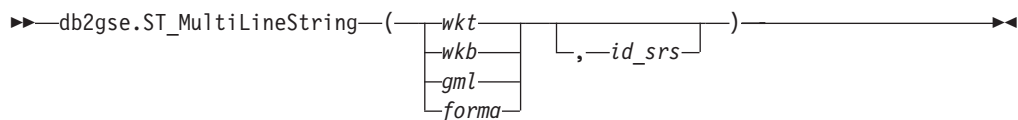
ST\_MultiLineString crea una multilínea a partir de uno de estos datos de entrada:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenida la multilínea resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma o la representación GML es nula, se devuelve un valor nulo.

### Sintaxis



### Parámetros

- wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la multilínea resultante.
- wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la multilínea resultante.
- gml** Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), de la multilínea resultante.
- forma** Valor de tipo BLOB(2G) que contiene la representación de forma de la multilínea resultante.
- id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la multilínea resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_MultiLineString

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_MultiLineString para crear e insertar una multilínea a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es una multilínea representada en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son:

- Línea 1: (33, 2) (34, 3) (35, 6)
- Línea 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Línea 3: (39, 3) (37, 4) (36, 7)

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER,
                             geometry ST_MultiLineString)
```

```
INSERT INTO sample_mlines
VALUES (1110,
        ST_MultiLineString ('multilineestring ( (33 2, 34 3, 35 6),
                                                (28 4, 29 5, 31 8, 43 12),
                                                (39 3, 37 4, 36 7) )', 1) )
```

La sentencia SELECT siguiente devuelve la multilínea que se registró en la tabla:

```
SELECT id,
        CAST( ST_AsText( geometry ) AS VARCHAR(280) )
        MULTI_LINE_STRING
FROM sample_mlines
WHERE id = 1110
```

Resultados:

```
ID          MULTI_LINE_STRING
-----
1110 MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000,
                        35.00000000 6.00000000),
                       ( 28.00000000 4.00000000, 29.00000000 5.00000000,
                        31.00000000 8.00000000, 43.00000000 12.00000000),
                       ( 39.00000000 3.00000000, 37.00000000 4.00000000,
                        36.00000000 7.00000000 ))
```

## ST\_MultiPoint

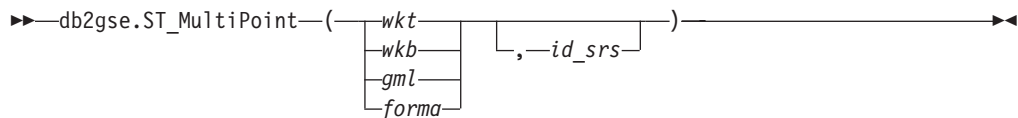
ST\_MultiPoint crea un multipunto a partir de uno de estos datos de entrada:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenido el multipunto resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma o la representación GML es nula, se devuelve un valor nulo.

### Sintaxis



### Parámetros

- wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional del multipunto resultante.
- wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional del multipunto resultante.
- gml** Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), del multipunto resultante.
- forma** Valor de tipo BLOB(2G) que contiene la representación de forma del multipunto resultante.
- id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipunto resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_Point

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_MultiPoint para crear e insertar un multipunto a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un multipunto representada en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son: (1, 2) (4, 3) (5, 6).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpoints (id INTEGER, geometry ST_MultiPoint)

INSERT INTO sample_mpoints
VALUES (1110, ST_MultiPoint ('multipoint (1 2, 4 3, 5 6) '), 1))
```

La sentencia SELECT siguiente devuelve el multipunto que se registró en la tabla:

```
SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(90)) MULTIPOINT
FROM sample_mpoints
WHERE id = 1110
```

Resultados:

```
ID          MULTIPOINT
-----
1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000
3.00000000, 5.00000000 6.00000000)
```

## ST\_MultiPolygon

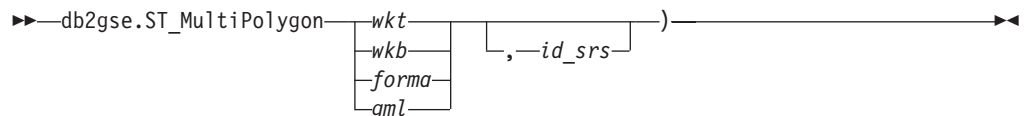
ST\_MultiPolygon crea un multipolígono a partir de uno de estos datos de entrada:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenido el multipolígono resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma o la representación GML es nula, se devuelve un valor nulo.

### Sintaxis



### Parámetros

- wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional del multipolígono resultante.
- wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional del multipolígono resultante.
- gml** Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), del multipolígono resultante.
- forma** Valor de tipo BLOB(2G) que contiene la representación de forma del multipolígono resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipolígono resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_MultiPolygon

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_MultiPolygon para crear e insertar un multipolígono a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un multipolígono representado en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son:

- Polígono 1: (3, 3) (4, 6) (5, 3) (3, 3)
- Polígono 2: (8, 24) (9, 25) (1, 28) (8, 24)
- Polígono 3: (13, 33) (7, 36) (1, 40) (10, 43) (13, 33)

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)
```

```
INSERT INTO sample_mpolys
VALUES (1110,
       ST_MultiPolygon ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                         (8 24, 9 25, 1 28, 8 24),
                                         (13 33, 7 36, 1 40, 10 43 13 33) ))', 1) )
```

La sentencia SELECT siguiente devuelve la multipolígono que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
FROM sample_mpolys
WHERE id = 1110
```

Resultados:

```
ID          MULTI_POLYGON
-----
1110 MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
1.00000000 40.00000000, 7.00000000 36.00000000,
13.00000000 33.00000000)),
(( 8.00000000 24.00000000, 9.00000000 25.00000000,
1.00000000 28.00000000, 8.00000000 24.00000000)),
(( 3.00000000 3.00000000, 4.00000000 6.00000000,
5.00000000 3.00000000, 3.00000000 3.00000000)))
```

---

## ST\_NumGeometries

ST\_NumGeometries toma una colección de geometrías como parámetro de entrada y devuelve el número de geometrías de la colección.



Si la colección de geometrías proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►—db2gse.ST\_NumGeometries—(—colección—)—————►

## Parámetro

### colección

Valor de tipo ST\_GeomCollection o uno de sus subtipos que representa la colección de geometrías para la que se obtiene el número de geometrías.

## Tipo de retorno

INTEGER

## Ejemplo

La tabla SAMPLE\_GEOMCOLL contiene dos colecciones de geometrías. Una es un multipolígono, y la otra es un multipunto. La función ST\_NumGeometries determina cuántas geometrías individuales están incluidas en cada colección de geometrías.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geomcoll (id INTEGER, geometry ST_GeomCollection)

INSERT INTO sample_geomcoll
VALUES( 1,
       ST_MultiPolygon ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                         (8 24, 9 25, 1 28, 8 24),
                                         (13 33, 7 36, 1 40, 10 43, 13 33) ))', 1) )

INSERT INTO sample_geomcoll
VALUES (2, ST_MultiPoint ('multipoint (1 2, 4 3, 5 6, 7 6, 8 8)', 1) )

SELECT id, ST_NumGeometries (geometry) NUM_GEOMS_IN_COLL
FROM sample_geomcoll
```

Resultados:

ID	NUM_GEOMS_IN_COLL
1	3
2	5

---

## ST\_NumInteriorRing

ST\_NumInteriorRing toma un polígono como parámetro de entrada y devuelve el número de sus anillos interiores.

Si el polígono proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

Si el polígono no tiene anillos interiores, se devuelve un 0 (cero).

También se puede invocar esta función como método.

## Sintaxis

►►—db2gse.ST\_NumInteriorRing—(—polígono—)—————►►

## Parámetro

### polígono

Valor de tipo ST\_Polygon que representa el polígono para el que se devuelve el número de anillos interiores.

## Tipo de retorno

INTEGER

## Ejemplo

El ejemplo siguiente crea dos polígonos:

- Un polígono con dos anillos interiores
- Un polígono sin anillos interiores

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1, ST_Polygon('polygon
((40 120, 90 120, 90 150, 40 150, 40 120),
(50 130, 60 130, 60 140, 50 140, 50 130),
(70 130, 80 130, 80 140, 70 140, 70 130))' , 0) )
```

```
INSERT INTO sample_polys
VALUES (2, ST_Polygon('polygon ((5 15, 50 15, 50 105, 5 15))' , 0) )
```

La función ST\_NumInteriorRing se utiliza para obtener el número de anillos de las geometrías contenidas en la tabla:

```
SELECT id, ST_NumInteriorRing(geometry) NUM_RINGS
FROM sample_polys
```

Resultados:

ID	NUM_RINGS
1	2
2	0

---

## ST\_NumLineStrings

ST\_NumLineStrings toma una multilínea como parámetro de entrada y devuelve el número de cadenas lineales de la multilínea.

Si la multilínea proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►►—db2gse.ST\_NumLineStrings—(—multilínea—)—————►►

## Parámetro

### multilínea

Valor de tipo ST\_MultiLineString que representa la multilínea para la que se devuelve el número de cadenas lineales.

## Tipo de retorno

INTEGER

## Ejemplo

La tabla SAMPLE\_MLINES contiene multilíneas. La función ST\_NumLineStrings determina cuántas geometrías individuales están incluidas en cada multilínea.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER, geometry ST_MultiLineString)
```

```
INSERT INTO sample_mlines
VALUES (110, ST_MultiLineString ('multilinestring
( (33 2, 34 3, 35 6),
(28 4, 29 5, 31 8, 43 12),
(39 3, 37 4, 36 7))', 1) )
INSERT INTO sample_mlines
VALUES (111, ST_MultiLineString ('multilinestring
( (3 2, 4 3, 5 6),
(8 4, 9 5, 3 8, 4 12))', 1) )
```

```
SELECT id, ST_NumLineStrings (geometry) NUM_WITHIN
FROM sample_mlines
```

Resultados:

ID	NUM_WITHIN
110	3
111	2

---

## ST\_NumPoints

ST\_NumPoints toma una geometría como parámetro de entrada y devuelve el número de puntos que se utilizaron para definir esa geometría. Por ejemplo, si la geometría es un polígono y se utilizaron cinco puntos para definir ese polígono, el número devuelto es 5.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►►db2gse.ST\_NumPoints(—*geometría*—)◀◀

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la que se obtiene el número de puntos.

## Tipo de retorno

INTEGER

## Ejemplo

La tabla contiene diversas geometrías. La función ST\_NumPoints determina cuántos puntos hay dentro de cada geometría contenida en la tabla SAMPLE\_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (spatial_type VARCHAR(18), geometry ST_Geometry)

INSERT INTO sample_geometries
VALUES ('st_point',
       ST_Point (2, 3, 0) )

INSERT INTO sample_geometries
VALUES ('st_linestring',
       ST_LineString ('linestring (2 5, 21 3, 23 10)', 0) )

INSERT INTO sample_geometries
VALUES ('st_polygon',
       ST_Polygon ('polygon ((110 120, 110 140, 120 130, 110 120))', 0) )

SELECT spatial_type, ST_NumPoints (geometry) NUM_POINTS
FROM sample_geometries
```

Resultados:

SPATIAL_TYPE	NUM_POINTS
st_point	1
st_linestring	3
st_polygon	4

---

## ST\_NumPolygons

ST\_NumPolygons toma un multipolígono como parámetro de entrada y devuelve el número de polígonos que contiene.

Si el multipolígono proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►►—db2gse.ST\_NumPolygons—(—*multipolígono*—)—————►►

## Parámetro

### multipolígono

Valor de tipo ST\_MultiPolygon que representa el multipolígono para el que se obtiene el número de polígonos.

## Tipo de retorno

INTEGER

## Ejemplo

La tabla SAMPLE\_MPOLYS contiene multipolígonos. La función ST\_NumPolygons determina cuántas geometrías individuales están incluidas en cada multipolígono.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)

INSERT INTO sample_mpolys
VALUES( 1,
       ST_MultiPolygon ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                         (8 24, 9 25, 1 28, 8 24),
                                         (13 33, 7 36, 1 40, 10 43, 13 33) ))', 1) )

INSERT INTO sample_polys
VALUES(2,
       ST_MultiPolygon ('multipolygon empty', 1) )

INSERT INTO sample_polys
VALUES (3,
       ST_MultiPolygon ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                         (13 33, 7 36, 1 40, 10 43, 13 33) ))', 1) )

SELECT id, ST_NumPolygons (geometry) NUM_WITHIN
FROM sample_mpolys
```

Resultados:

ID	NUM_WITHIN
1	3
2	0
3	2

---

## ST\_Overlaps

ST\_Overlaps toma dos geometrías como parámetros de entrada y devuelve un 1 si la intersección de las geometrías es una geometría de la misma dimensión pero que es diferente de las geometrías originales. En caso contrario, se devuelve un 0 (cero).

Si cualquiera de las dos geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

### Sintaxis

```
db2gse.ST_Overlaps(geometría1, geometría2)
```

### Parámetros

#### *geometría1*

Un valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría que se comprueba si se solapa con *geometría2*.

#### *geometría2*

Un valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría que se comprueba si se solapa con *geometría1*.

## Tipo de retorno

INTEGER

## Ejemplos

### Ejemplo 1

En este ejemplo, se muestra el uso de ST\_Overlaps. Se crean varias geometrías y luego se insertan en la tabla SAMPLE\_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)
```

```
INSERT INTO sample_geometries
VALUES (1, ST_Point (10, 20, 1)),
      (2, ST_Point ('point (41 41)', 1) ),
      (10, ST_LineString ('linestring (1 10, 3 12, 10 10)', 1) ),
      (20, ST_LineString ('linestring (50 10, 50 12, 45 10)', 1) ),
      (30, ST_LineString ('linestring (50 12, 50 10, 60 8)', 1) ),
      (100, ST_Polygon ('polygon ((0 0, 0 40, 40 40, 40 0, 0 0))', 1) ),
      (110, ST_Polygon ('polygon ((30 10, 30 30, 50 30, 50 10, 30 10))', 1) ),
      (120, ST_Polygon ('polygon ((0 50, 0 60, 40 60, 40 60, 0 50))', 1) )
```

### Ejemplo 2

Este ejemplo determina los ID de puntos que se solapan.

```
SELECT sg1.id, sg2.id
CASE ST_Overlaps (sg1.geometry, sg2.geometry)
  WHEN 0 THEN 'Points_do_not_overlap'
  WHEN 1 THEN 'Points_overlap'
END
AS OVERLAP
FROM sample_geometries sg1, sample_geometries sg2
WHERE sg1.id < 10 AND sg2.id < 10 AND sg1.id >= sg2.id
```

Resultados:

ID	ID	OVERLAP
	1	1 Points_do_not_overlap
	2	1 Points_do_not_overlap
	2	2 Points_do_not_overlap

### Ejemplo 3

Este ejemplo determina los ID de líneas que se solapan.

```
SELECT sg1.id, sg2.id
CASE ST_Overlaps (sg1.geometry, sg2.geometry)
  WHEN 0 THEN 'Lines_do_not_overlap'
  WHEN 1 THEN 'Lines_overlap'
END
AS OVERLAP
FROM sample_geometries sg1, sample_geometries sg2
WHERE sg1.id >= 10 AND sg1.id < 100
  AND sg2.id >= 10 AND sg2.id < 100
  AND sg1.id >= sg2.id
```

Resultados:

ID	ID	OVERLAP
	10	10 Lines_do_not_overlap
	20	10 Lines_do_not_overlap

```

30      10 Lines_do_not_overlap
20      20 Lines_do_not_overlap
30      20 Lines_overlap
30      30 Lines_do_not_overlap

```

#### Ejemplo 4

Este ejemplo determina los ID de polígonos que se solapan.

```

SELECT sg1.id, sg2.id
  CASE ST_Overlaps (sg1.geometry, sg2.geometry)
    WHEN 0 THEN 'Polygons_do_not_overlap'
    WHEN 1 THEN 'Polygons_overlap'
  END
  AS OVERLAP
FROM sample_geometries sg1, sample_geometries sg2
WHERE sg1.id >= 100 AND sg2.id >= 100 AND sg1.id >= sg2.id

```

Resultados:

ID	ID	OVERLAP
100	100	Polygons_do_not_overlap
110	100	Polygons_overlap
120	100	Polygons_do_not_overlap
110	110	Polygons_do_not_overlap
120	110	Polygons_do_not_overlap
120	120	Polygons_do_not_overlap

## ST\_Perimeter

ST\_Perimeter toma como parámetros de entrada una superficie o multisuperficie y, opcionalmente, una unidad de medida, y devuelve el perímetro de la superficie o multisuperficie, que es la longitud de su perímetro, expresado en la unidad de medida proporcionada o en la unidad por omisión.

Si la superficie o multisuperficie proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```

▶▶ db2gse.ST_Perimeter (—superficie— [,—unidad—]) ▶▶

```

### Parámetros

#### superficie

Valor de tipo ST\_Surface, ST\_MultiSurface o uno de sus subtipos para el que se obtiene el perímetro.

#### unidad

Valor de tipo VARCHAR(128) que identifica las unidades utilizadas para medir el perímetro. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST\_UNITS\_OF\_MEASURE.

Si se omite el parámetro *unidad*, se siguen las normas siguientes para determinar la unidad empleada para medir el perímetro:

- Si *superficie* está en un sistema de coordenadas proyectadas o geocéntricas, la unidad lineal asociada a este sistema de coordenadas es el valor por omisión.
- Si *superficie* está en un sistema de coordenadas geográficas, pero no está en un sistema de referencia espacial (SRS) geodésico, la unidad angular asociada a este sistema de coordenadas es el valor por omisión.
- Si *superficie* está en un SRS geodésico, la unidad de medida por omisión es el metro.

**Restricciones para las conversiones de unidades:** Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La geometría está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La geometría está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La geometría está en un sistema de coordenadas geográficas, pero no está en un SRS geodésico y se especifica una unidad lineal.
- La geometría está en un sistema de coordenadas geográficas, está en un SRS geodésico y se especifica una unidad lineal.

## Tipo de retorno

DOUBLE

## Ejemplos

### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_Perimeter. Se invoca a db2se para crear un sistema de referencia espacial cuyo ID es 4000, y se crea un polígono en ese sistema de referencia espacial.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
```

```
db2se create_srs se_bank -srsId 4000 -srsName new_york1983
  -xOffset 0 -yOffset 0 -xScale 1 -yScale 1
  -coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet
```

La tabla SAMPLE\_POLYS se crea para contener una geometría cuyo perímetro es 18.

```
CREATE TABLE sample_polys (id SMALLINT, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon ('polygon ((0 0, 0 4, 5 4, 5 0, 0 0))', 4000))
```

### Ejemplo 2

Este ejemplo obtiene el ID y el perímetro del polígono.

```
SELECT id, ST_Perimeter (geometry) AS PERIMETER
  FROM sample_polys
```

Resultados:

```
ID          PERIMETER
-----
1          +1.800000000000000E+001
```

### Ejemplo 3



Este ejemplo obtiene el ID y el perímetro del polígono, con el perímetro expresado en metros.

```
SELECT id, ST_Perimeter (geometry, 'METER') AS PERIMETER_METER
FROM sample_polys
```

Resultados:

ID	PERIMETER_METER
1	+5.48641097282195E+000

---

## ST\_PerpPoints

ST\_PerpPoints toma como parámetros de entrada una curva o multicurva y un punto y devuelve la proyección perpendicular de ese punto en la curva o multicurva. Se obtiene el punto con la distancia más pequeña entre el punto proporcionado y el punto perpendicular. Si dos o más puntos proyectados perpendicularmente son equidistantes respecto al punto proporcionado, se devuelven todos ellos. Si no puede crearse ningún punto perpendicular, se devuelve un punto vacío.

Si la curva o multicurva proporcionada tiene coordenadas Z o M, la coordenada Z o M de los puntos resultantes se calcula por interpolación sobre la curva o multicurva proporcionada.

Si la curva o punto proporcionados están vacíos, se devuelve un punto vacío. Si la curva o punto proporcionados es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_PerpPoints—(—*curva*—,—*punto*—)—————►►

### Parámetros

**curva** Valor de tipo ST\_Curve, ST\_MultiCurve o uno de sus subtipos que representa la curva o multicurva en la que se obtiene la proyección perpendicular del *punto*.

**punto** Un valor de tipo ST\_Point que representa el punto que se proyecta perpendicularmente sobre *curva*.

### Tipo de retorno

db2gse.ST\_MultiPoint

### Ejemplos

#### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_PerpPoints para averiguar los puntos que son perpendiculares a la cadena lineal almacenada en la siguiente tabla. La función ST\_LineString se utiliza en la sentencia INSERT para crear la cadena lineal.

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)

INSERT INTO sample_lines (id, line)
VALUES (1, ST_LineString('linestring (0 10, 0 0, 10 0, 10 10)' , 0) )

```

## Ejemplo 2

Este ejemplo obtiene la proyección perpendicular sobre la cadena lineal de un punto cuyas coordenadas son (5, 0). La función ST\_AsText se utiliza para convertir el valor devuelto (un multipunto) en su correspondiente representación de texto convencional.

```

SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 0) ) )
AS VARCHAR(50) ) PERP
FROM sample_lines

```

Resultados:

```

PERP
-----
MULTIPOINT ( 5.00000000 0.00000000)

```

## Ejemplo 3

Este ejemplo obtiene la proyección perpendicular sobre la cadena lineal de un punto cuyas coordenadas son (5, 5). En este caso, existen tres puntos en la cadena lineal que son equidistantes de la ubicación proporcionada. Por tanto, se obtiene un multipunto formado por los tres puntos.

```

SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 5) ) )
AS VARCHAR(160) ) PERP
FROM sample_lines

```

Resultados:

```

PERP
-----
MULTIPOINT ( 0.00000000 5.00000000, 5.00000000 0.00000000, 10.00000000 5.00000000)

```

## Ejemplo 4

Este ejemplo obtiene la proyección perpendicular sobre la cadena lineal de un punto cuyas coordenadas son (5, 10). En este caso, se pueden encontrar tres puntos perpendiculares diferentes. Pero la función ST\_PerpPoints sólo devuelve los puntos que están más cerca del punto proporcionado. Por tanto, se obtiene un multipunto formado solamente por los dos puntos más cercanos. El tercer punto no se incluye.

```

SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 10) ) )
AS VARCHAR(80) ) PERP
FROM sample_lines

```

Resultados:

```

PERP
-----
MULTIPOINT ( 0.00000000 10.00000000, 10.00000000 10.00000000 )

```

## Ejemplo 5

Este ejemplo obtiene la proyección perpendicular sobre la cadena lineal de un punto cuyas coordenadas son (5, 15).

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point('point(5 15)', 0 ) ) )
AS VARCHAR(80) ) PERP
FROM sample_lines
```

Resultados:

```
PERP
-----
MULTIPOINT ( 5.00000000 0.00000000 )
```

### Ejemplo 6

En este ejemplo, el punto especificado cuyas coordenadas son (15 15) carece de proyección perpendicular sobre la cadena lineal. Por tanto, se devuelve una geometría vacía.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(15, 15) ) )
AS VARCHAR(80) ) PERP
FROM sample_lines
```

Resultados:

```
PERP
-----
MULTIPOINT EMPTY
```

## ST\_Point

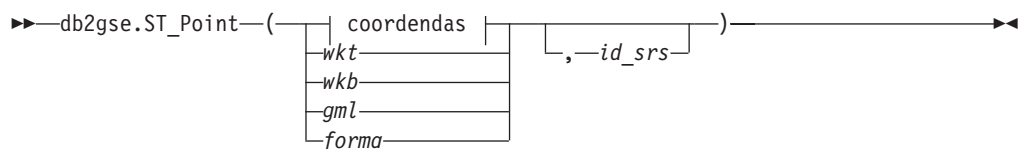
ST\_Point obtiene un punto a partir de uno de estos conjuntos de datos de entrada:

- Coordenadas X e Y solamente
- Coordenadas X, Y y Z
- Coordenadas X, Y, Z y M
- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

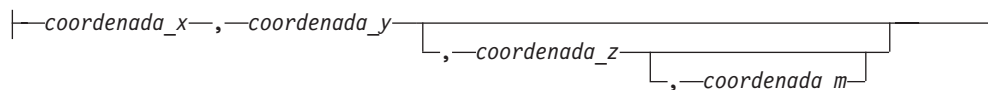
Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenido el punto resultante.

Si el punto se crea a partir de coordenadas, y si la coordenada X o Y es nula, se emite una condición de excepción (SQLSTATE 38SUP). Si la coordenada Z o M es nula, el punto resultante no tendrá coordenada Z o M, respectivamente. Si el punto se crea a partir de su representación de texto convencional, su representación binaria convencional, su representación de forma o su representación GML, y esta representación es nula, se devuelve un valor nulo.

### Sintaxis



**coordenadas:**



**Parámetros**

- wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional del punto resultante.
- wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional del punto resultante.
- gml** Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), del punto resultante.
- forma** Valor de tipo BLOB(2G) que contiene la representación de forma del punto resultante.
- id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al punto resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo `DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS`, se emite una condición de excepción (`SQLSTATE 38SU1`).

**coordenada\_x**

Valor de tipo DOUBLE que especifica la coordenada X del punto resultante.

**coordenada\_y**

Valor de tipo DOUBLE que especifica la coordenada Y del punto resultante.

**coordenada\_z**

Valor de tipo DOUBLE que especifica la coordenada Z del punto resultante.

Si se omite el parámetro *coordenada\_z*, el punto resultante no tendrá coordenada Z. Para ese punto, el resultado de `ST_Is3D` será un 0 (cero).

**coordenada\_m**

Valor de tipo DOUBLE que especifica la coordenada M del punto resultante.

Si se omite el parámetro *coordenada\_m*, el punto resultante no tendrá una medida. Para ese punto, el resultado de `ST_IsMeasured` será un 0 (cero).

**Tipo de retorno**

`db2gse.ST_Point`

**Ejemplo**

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

### Ejemplo 1

Este ejemplo muestra el uso de ST\_Point para crear e insertar puntos. El primer punto se crea utilizando un conjunto de coordenadas X e Y. El segundo punto se crea utilizando su representación de texto convencional. Ambos puntos son geometrías que están contenidas en el sistema de referencia espacial 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
```

```
INSERT INTO sample_points
VALUES (1100, ST_Point (10, 20, 1) )
```

```
INSERT INTO sample_points
VALUES (1101, ST_Point ('point (30 40)', 1) )
```

La sentencia SELECT siguiente obtiene los puntos que se registraron en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90)) POINTS
FROM sample_points
```

Resultados:

ID	POINTS
1100	POINT ( 10.00000000 20.00000000)
1101	POINT ( 30.00000000 40.00000000)

### Ejemplo 2

Este ejemplo inserta un registro en la tabla SAMPLE\_POINTS cuyo ID es 1103 y un valor de punto que tiene una coordenada X igual a 120, una coordenada Y igual a 358, una coordenada M igual a 34, y carece de coordenada Z.

```
INSERT INTO SAMPLE_POINTS(ID, GEOMETRY)
VALUES(1103, db2gse.ST_Point(120, 358, CAST(NULL AS DOUBLE), 34, 1))
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90) ) POINTS
FROM sample_points
```

Resultados:

ID	POINTS
1103	POINT M ( 120.00000000 358.00000000 34.00000000)

### Ejemplo 3

Este ejemplo inserta una fila en la tabla SAMPLE\_POINTS cuyo ID es 1104 y un valor de punto que tiene una coordenada X igual a 1003, una coordenada Y igual a 9876, una coordenada Z igual a 20, en el sistema de referencia espacial 0, y utilizando la representación GML.

```
INSERT INTO SAMPLE_POINTS(ID, GEOMETRY)
VALUES(1104, db2gse.ST_Point('<gml:Point><gml:coord>
<gml:x>1003</gml:x><gml:y>9876</gml:y><gml:z>20</gml:z>
</gml:coord></gml:Point>', 1))
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90) ) POINTS
FROM sample_points
```

Resultados:

ID	POINTS
1104	POINT Z ( 1003.000000 9876.000000 20.00000000)

---

## ST\_PointAtDistance

ST\_PointAtDistance toma una geometría de curva o multicurva y una distancia como parámetros de entrada y devuelve la geometría de punto en la distancia dada sobre la geometría de curva.

También se puede invocar esta función como método.

### Sintaxis

► db2gse.ST\_PointAtDistance(*—geometría—*, *—distancia—*) ◀

### Parámetro

#### geometría

Valor de tipo ST\_Curve o ST\_MultiCurve o uno de sus subtipos que representa la geometría que debe procesarse.

#### distancia

Valor de tipo DOUBLE que representa la distancia sobre la geometría para localizar el punto.

### Tipo de retorno

db2gse.ST\_Point

### Ejemplo

#### Ejemplo 1

La sentencia de SQL siguiente crea la tabla SAMPLE\_GEOMETRIES con dos columnas. La columna de ID identifica exclusivamente cada fila. La columna GEOMETRY ST\_LineString almacena geometrías de ejemplo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

La sentencia de SQL siguiente inserta dos filas en la tabla SAMPLE\_GEOMETRIES.

```
INSERT INTO sample_geometries(id, geometry)
VALUES
(1,ST_LineString('LINESTRING ZM(0 0 0 0, 10 100 1000 10000)',1)),
(2,ST_LineString('LINESTRING ZM(10 100 1000 10000, 0 0 0 0)',1))
```

La sentencia SELECT siguiente y el conjunto de resultados correspondiente muestra cómo usar la función ST\_PointAtDistance para buscar puntos a una distancia de 15 unidades de coordenada a partir del principio de la cadena lineal.

```
SELECT ID, VARCHAR(ST_AsText(ST_PointAtDistance(geometry, 15)), 50) AS POINTAT
FROM sample_geometries
ID POINTAT
```

```
-----
1 POINT ZM(1.492556 14.925558 149 1493)
2 POINT ZM(8.507444 85.074442 851 8507)
```

2 registro(s) seleccionados.

---

## ST\_PointFromText

ST\_PointFromText toma como parámetros de entrada una representación de texto convencional de un punto y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el punto correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_Point. Es recomendable debido a su flexibilidad: ST\_Point acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

### Sintaxis

```
db2gse.ST_PointFromText( ( wkt [, id_srs ] ) )
```

### Parámetros

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional del punto resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al punto resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_Point

### Ejemplo

Este ejemplo muestra cómo utilizar ST\_PointFromText para crear e insertar un punto a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un punto representado en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son: (10, 20).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
```

```
INSERT INTO sample_points
VALUES (1110, ST_PointFromText ('point (30 40)', 1) )
```

La sentencia SELECT siguiente devuelve el polígono que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(35) ) POINTS
FROM sample_points
WHERE id = 1110
```

Resultados:

ID	POINTS
-----	
	1110 POINTS ( 30.00000000 40.00000000)

## ST\_PointFromWKB

ST\_PointFromWKB toma como parámetros de entrada una representación binaria convencional de un punto y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el punto correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_Point. Es recomendable debido a su flexibilidad: ST\_Point acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

### Sintaxis

```
db2gse.ST_PointFromWKB(wkb, id_srs)
```

### Parámetros

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional del punto resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al punto resultante.

Si se omite el parámetro *id\_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_Point

### Ejemplo

Este ejemplo muestra cómo utilizar ST\_PointFromWKB para crear un punto a partir de su representación binaria convencional. Las geometrías son puntos en el sistema de referencia espacial 1. En este ejemplo, los puntos se guardan en la columna GEOMETRY de la tabla SAMPLE\_POLYS, y luego la columna WKB se actualiza con su representación binaria convencional (utilizando la función ST\_AsBinary). Finalmente, se utiliza la función ST\_PointFromWKB para obtener los puntos a partir de la columna WKB.

La tabla SAMPLE\_POINTS tiene una columna GEOMETRY, donde se guardan los puntos, y una columna WKB, donde se guardan las representaciones binarias convencionales de los puntos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point, wkb BLOB(32K))
```



```

INSERT INTO sample_points
VALUES (10, ST_Point ('point (44 14)', 1) ),
VALUES (11, ST_Point ('point (24 13)', 1))

UPDATE sample_points AS temporary_correlated
SET wkb = ST_AsBinary( geometry )
WHERE id = temporary_correlated.id

```

En la sentencia SELECT siguiente, se utiliza la función ST\_PointFromWKB para obtener los puntos a partir de la columna WKB.

```

SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) ) AS VARCHAR(35) ) POINTS
FROM sample_points

```

Resultados:

ID	POINTS
10	POINT ( 44.00000000 14.00000000)
11	POINT ( 24.00000000 13.00000000)

## ST\_PointN

ST\_PointN toma una cadena lineal o un multipunto, y un índice como parámetros de entrada y devuelve el punto identificado por el índice y perteneciente a la cadena lineal o multipunto. El punto resultante se representa según el sistema de referencia espacial de la cadena lineal o multipunto proporcionados.

Si la cadena lineal o multipunto proporcionados es un valor nulo o está vacío, se devuelve un valor nulo. Si el índice es menor que 1 o mayor que el número de puntos de la cadena lineal o multipunto, se devuelve un valor nulo y se emite una condición de aviso (SQLSTATE 01HS2).

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_PointN—(—geometría—,—índice—)—————►►

### Parámetros

#### geometría

Un valor de tipo ST\_LineString o ST\_MultiPoint que representa la geometría a partir de la que se devuelve el punto que está identificado mediante *índice*.

**índice** Un valor de tipo INTEGER que identifica el punto número *n* que se obtiene de *geometría*.

### Tipo de retorno

db2gse.ST\_Point

### Ejemplo

El ejemplo siguiente muestra el uso de ST\_PointN.

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)

```

```

INSERT INTO sample_lines
VALUES (1, ST_LineString ('linestring (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0) )

SELECT id, CAST ( ST_AsText (ST_PointN (line, 2) ) AS VARCHAR(60) ) SECOND_INDEX
FROM sample_lines

```

Resultados:

```

ID          SECOND_INDEX
-----
1 POINT (5.00000000 5.00000000)

```

---

## ST\_PointOnSurface

ST\_PointOnSurface toma como parámetro de entrada una superficie o multisuperficie y devuelve un punto que con toda seguridad estará en el interior de la superficie o multisuperficie. Este punto es el paracentro de la superficie.

El punto resultante se representa según el sistema de referencia espacial de la superficie o multisuperficie proporcionada.

Si la superficie o multisuperficie proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_PointOnSurface—(—superficie—)—————►►

### Parámetro

#### superficie

Valor de tipo ST\_Surface, ST\_MultiSurface o uno de sus subtipos que representa la geometría para la que se obtiene un punto situado en la superficie.

### Tipo de retorno

db2gse.ST\_Point

### Ejemplo

El ejemplo siguiente crea dos polígonos y luego utiliza la función ST\_PointOnSurface. Uno de los polígonos tiene un hueco en su centro. Los puntos obtenidos están situados en la superficie de los polígonos. No se encuentran necesariamente en el centro exacto de los polígonos.

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
VALUES( 1,
        ST_Polygon ('polygon ( (40 120, 90 120, 90 150, 40 150, 40 120) ,
                             (50 130, 80 130, 80 140, 50 140, 50 130) )' ,0) )

INSERT INTO sample_polys
VALUES(2,
        ST_Polygon ('polygon ( (10 10, 50 10, 10 30, 10 10) )', 0) )

```

```
SELECT id, CAST (ST_AsText (ST_PointOnSurface (geometry) ) AS VARCHAR(80) )
      POINT_ON_SURFACE
FROM sample_polys
```

Resultados:

```
ID          POINT_ON_SURFACE
-----
1 POINT ( 65.00000000 125.00000000)
2 POINT ( 30.00000000 15.00000000)
```

---

## ST\_PolyFromText

ST\_PolyFromText toma como parámetros de entrada una representación de texto convencional de un polígono y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el polígono correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_Polygon. Es recomendable debido a su flexibilidad: ST\_Polygon acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

### Sintaxis

```
db2gse.ST_PolyFromText (wkt [, id_srs])
```

### Parámetros

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional del polígono resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al polígono resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_Polygon

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST\_PolyFromText para crear e insertar un polígono a partir de su representación de texto convencional. El registro que se

inserta tiene un ID = 1110, y la geometría es un polígono representado en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son: (50, 20) (50, 40) (70, 30).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1110, ST_PolyFromText ('polygon ((50 20, 50 40, 70 30, 50 20))', 1) )
```

La sentencia SELECT siguiente devuelve el polígono que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGON
FROM sample_polys
WHERE id = 1110
```

Resultados:

```
ID          POLYGON
-----
1110 POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000,
                50.00000000 40.00000000, 50.00000000 20.00000000))
```

---

## ST\_PolyFromWKB

ST\_PolyFromWKB toma como parámetros de entrada una representación binaria convencional de un polígono y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el polígono correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST\_Polygon. Es recomendable debido a su flexibilidad: ST\_Polygon acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

### Sintaxis

```
db2gse.ST_PolyFromWKB(wkb [, id_srs])
```

### Parámetros

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional del polígono resultante.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al polígono resultante.

Si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

### Tipo de retorno

db2gse.ST\_Polygon

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar `ST_PolyFromWKB` para crear un polígono a partir de su representación binaria convencional. La geometría es un polígono en el sistema de referencia espacial 1. En este ejemplo, el polígono se guarda con el ID = 1115 en la columna `GEOMETRY` de la tabla `SAMPLE_POLYS`, y luego la columna `WKB` se actualiza con su representación binaria convencional (utilizando la función `ST_AsBinary`). Finalmente, se utiliza la función `ST_PolyFromWKB` para obtener el multipolígono a partir de la columna `WKB`. Las coordenadas X e Y de esta geometría son: (50, 20) (50, 40) (70, 30).

La tabla `SAMPLE_POLYS` tiene una columna `GEOMETRY`, donde se guarda el polígono, y una columna `WKB`, donde se guarda la representación binaria convencional de polígono.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon,
    wkb BLOB(32K))

INSERT INTO sample_polys
    VALUES (10, ST_Polygon ('polygon ((50 20, 50 40, 70 30, 50 20))', 1) )

UPDATE sample_polys AS temporary_correlated
    SET wkb = ST_AsBinary( geometry )
    WHERE id = temporary_correlated.id
```

En la sentencia `SELECT` siguiente, se utiliza la función `ST_PolyFromWKB` para obtener el polígono a partir de la columna `WKB`.

```
SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) )
    AS VARCHAR(120) ) POLYGON
    FROM sample_polys
    WHERE id = 1115
```

Resultados:

```
ID          POLYGON
-----
1115 POLYGON (( 50.00000000 20.00000000, 70.00000000
    30.00000000,50.00000000 40.00000000, 50.00000000
    20.00000000))
```

---

## ST\_Polygon

`ST_Polygon` crea un polígono a partir de uno de estos datos de entrada:

- Una cadena lineal cerrada que define el anillo exterior del polígono resultante
- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

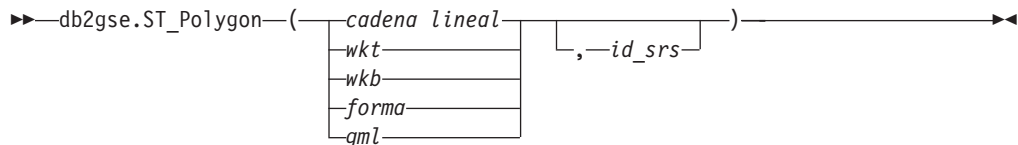
Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenido el polígono resultante.

Si el polígono se crea a partir de una cadena lineal y ésta es un valor nulo, se devuelve un valor nulo. Si la cadena lineal proporcionada está vacía, se obtiene un

polígono vacío. Si el polígono se crea a partir de su representación de texto convencional, su representación binaria convencional, su representación de forma o su representación GML, y la representación es nula, se devuelve un valor nulo.

También se puede invocar esta función como método para los casos siguientes solamente: `ST_Polygon(cadena lineal)` y `ST_Polygon(cadena lineal, id_srs)`.

## Sintaxis



## Parámetros

### cadena lineal

Valor de tipo `ST_LineString` que representa la cadena lineal que define el anillo exterior correspondiente al perímetro externo. Si *cadena lineal* no es cerrada y es simple, se emite una condición de excepción (SQLSTATE 38SSSL).

**wkt** Valor de tipo `CLOB(2G)` que contiene la representación de texto convencional del polígono resultante.

**wkb** Valor de tipo `BLOB(2G)` que contiene la representación binaria convencional del polígono resultante.

**forma** Valor de tipo `BLOB(2G)` que contiene la representación de forma del polígono resultante.

**gml** Valor de tipo `CLOB(2G)` que contiene la representación, en formato GML (Geography Markup Language), del polígono resultante.

**id\_srs** Valor de tipo `INTEGER` que identifica el sistema de referencia espacial correspondiente al polígono resultante.

Si el polígono se crea a partir del parámetro *cadena\_lineal* y se omite el parámetro *id\_srs*, se utiliza implícitamente el sistema de referencia espacial de *cadena\_lineal*. En otro caso, si se omite el parámetro *id\_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo `DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS`, se emite una condición de excepción (SQLSTATE 38SU1).

## Tipo de retorno

`db2gse.ST_Polygon`

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de ST\_Polygon para crear e insertar polígonos. Se crean e insertan tres polígonos. Todos ellos son geometrías que están contenidas en el sistema de referencia espacial 1.

- El primer polígono se crea a partir de un anillo (una cadena lineal cerrada y simple). Las coordenadas X e Y de este polígono son: (10, 20) (10, 40) (20, 30).
- El segundo polígono se crea utilizando su representación de texto convencional. Las coordenadas X e Y de este polígono son: (110, 120) (110, 140) (120, 130).
- El tercer polígono es un polígono con un hueco en su centro. Un polígono con un hueco en el centro consta de un polígono interior y un polígono exterior. Este polígono con un hueco en el centro se crea utilizando su representación de texto convencional. Las coordenadas X e Y del polígono exterior son: (110, 120) (110, 140) (130, 140) (130, 120) (110, 120). Las coordenadas X e Y del polígono interior son: (115, 125) (115, 135) (125, 135) (125, 135) (115, 125).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
VALUES (1100,
       ST_Polygon (ST_LineString ('linestring
                                (10 20, 10 40, 20 30, 10 20)',1), 1))
```

```
INSERT INTO sample_polys
VALUES (1101,
       ST_Polygon ('polygon
                   ((110 120, 110 140, 120 130, 110 120))', 1))
```

```
INSERT INTO sample_polys
VALUES (1102,
       ST_Polygon ('polygon
                   ((110 120, 110 140, 130 140, 130 120, 110 120),
                    (115 125, 115 135, 125 135, 125 135, 115 125))', 1))
```

La sentencia SELECT siguiente obtiene los polígonos que se registraron en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGONS
FROM sample_polys
```

Resultados:

ID	POLYGONS
1110	POLYGON (( 10.00000000 20.00000000, 20.00000000 30.00000000 10.00000000 40.00000000, 10.00000000 20.00000000))
1101	POLYGON (( 110.00000000 120.00000000, 120.00000000 130.00000000 110.00000000 140.00000000, 110.00000000 120.00000000))
1102	POLYGON (( 110.00000000 120.00000000, 130.00000000 120.00000000 130.00000000 140.00000000, 110.00000000 140.00000000 110.00000000 120.00000000), ( 115.00000000 125.00000000, 115.00000000 135.00000000 125.00000000 135.00000000, 125.00000000 135.00000000 115.00000000 125.00000000))

## ST\_PolygonN

ST\_PolygonN toma un multipolígono y un índice como parámetros de entrada y devuelve el polígono identificado por el índice. El polígono resultante se representa según el sistema de referencia espacial del multipolígono proporcionado.

Si el multipolígono proporcionado es nulo o está vacío, o si el índice es menor que 1 o mayor que el número de polígonos, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►►—db2gse.ST\_PolygonN—(—multipolígono—,—índice—)————►►

## Parámetros

### multipolígono

Valor de tipo ST\_MultiPolygon que representa el multipolígono a partir del cual se obtiene el polígono identificado por *índice*.

**índice** Un valor de tipo INTEGER que identifica el polígono número *n* que se debe obtener de *multipolígono*.

## Tipo de retorno

db2gse.ST\_Polygon

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de ST\_PolygonN.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)

INSERT INTO sample_mpolys
VALUES (1, ST_Polygon ('multipolygon (((3 3, 4 6, 5 3, 3 3),
                                     (8 24, 9 25, 1 28, 8 24)
                                     (13 33, 7 36, 1 40, 10 43,
                                     13 33)))', 1))

SELECT id, CAST ( ST_AsText (ST_PolygonN (geometry, 2) )
AS VARCHAR(120) ) SECOND_INDEX
FROM sample_mpolys
```

Resultados:

```
ID          SECOND_INDEX
-----
1 POLYGON (( 8.00000000 24.00000000, 9.00000000 25.00000000,
             1.00000000 28.00000000, 8.00000000 24.00000000))
```

---

## ST\_Relate

ST\_Relate toma dos geometrías y una matriz DE-9IM como parámetros de entrada y devuelve 1 si dichas geometrías cumplen las condiciones especificadas por la matriz. En caso contrario, se devuelve un 0 (cero).

Si cualquiera de las geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.



Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

También se puede invocar esta función como método.

## Sintaxis

```
► db2gse.ST_Relate(—geometría1—,—geometría2—,—matriz—) ◀
```

## Parámetros

### geometría1

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se compara con *geometría2*.

### geometría2

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se compara con *geometría1*.

**matriz** Valor de tipo CHAR(9) que representa la matriz DE-9IM (Dimensionally Extended 9 Intersection Model) que se utilizará para la verificación de *geometría1* y *geometría2*.

## Tipo de retorno

INTEGER

## Ejemplo

El ejemplo siguiente crea dos polígonos separados. Luego, se utiliza la función ST\_Relate para determinar diversas relaciones entre los dos polígonos. Por ejemplo, se determina si los dos polígonos se solapan.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
VALUES( 1,
       ST_Polygon('polygon ( (40 120, 90 120, 90 150, 40 150, 40 120) )', 0))
INSERT INTO sample_polys
VALUES(2,
       ST_Polygon('polygon ( (30 110, 50 110, 50 130, 30 130, 30 110) )', 0))

SELECT ST_Relate(a.geometry, b.geometry, 'T*T**T**') "Overlaps ",
       ST_Relate(a.geometry, b.geometry, 'T*T***FF*') "Contains ",
       ST_Relate(a.geometry, b.geometry, 'T*F***F***') "Within ",
       ST_Relate(a.geometry, b.geometry, 'T*****') "Intersects",
       ST_Relate(a.geometry, b.geometry, 'T*F***FF2') "Equals "
FROM sample_polys a, sample_polys b
WHERE a.id = 1 AND b.id = 2
```

Resultados:

Overlaps	Contains	Within	Intersects	Equals
1	0	0	1	0

---

## ST\_RemovePoint

ST\_RemovePoint toma una curva y un punto como parámetros de entrada y devuelve dicha curva después de eliminar de ella todos los puntos que son iguales al punto especificado. Si la curva proporcionada tiene coordenadas Z o M, el punto también debe tener coordenadas Z o M. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si la curva proporcionada está vacía, se devuelve una curva vacía. Si la curva proporcionada es un valor nulo, o si el punto proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►► db2gse.ST\_RemovePoint(—*curva*—, —*punto*—) ◀◀

### Parámetros

**curva** Valor de tipo ST\_Curve o uno de sus subtipos que representa la curva de la cual se elimina *punto*.

**punto** Valor de tipo ST\_Point que identifica los puntos que se eliminan de *curva*.

### Tipo de retorno

db2gse.ST\_Curve

### Ejemplos

#### Ejemplo 1

El ejemplo siguiente añade dos cadenas lineales a la tabla SAMPLE\_LINES. Estas cadenas lineales se utilizan en los ejemplos mostrados más abajo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)
```

```
INSERT INTO sample_lines
VALUES (1, ST_LineString('linestring
(10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0))
```

```
INSERT INTO sample_lines
VALUES (2, ST_LineString('linestring z
(0 0 4, 5 5 5, 10 10 6, 5 5 7, 0 0 8)', 0))
```

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

#### Ejemplo 2

El ejemplo siguiente elimina el punto (5, 5) de la cadena lineal cuyo ID es 1. Este punto aparece dos veces en la cadena lineal. Por tanto, se eliminan ambas apariciones.

```
SELECT CAST(ST_AsText (ST_RemovePoint (line, ST_Point(5, 5) ) )
AS VARCHAR(120) ) RESULT
FROM sample_lines
WHERE id = 1
```

Resultados:

```
RESULT
-----
LINESTRING ( 10.00000000 10.00000000, 0.00000000 0.00000000,
            10.00000000 0.00000000, 0.00000000 10.00000000)
```

### Ejemplo 3

El ejemplo siguiente elimina el punto (5, 5, 5) de la cadena lineal cuyo ID es 2. Este punto aparece una sola vez, por lo que sólo se elimina esa aparición.

```
SELECT CAST (ST_AsText (ST_RemovePoint (line, ST_Point(5.0, 5.0, 5.0)))
AS VARCHAR(160) ) RESULT
FROM sample_lines
WHERE id=2
```

Resultados:

```
RESULT
-----
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 10.00000000 10.00000000
            6.00000000, 5.00000000 5.00000000 7.00000000, 0.00000000
            0.00000000 8.00000000)
```

---

## ST\_SrsId, ST\_SRID

ST\_SrsId (o ST\_SRID) toma como parámetros de entrada una geometría y, opcionalmente, un identificador de sistemas de referencia espacial. Los datos devueltos dependen de los parámetros de entrada que se hayan especificado:

- Si se especifica el identificador de sistemas de referencia espacial, la función devuelve la geometría con su identificador cambiado al identificador especificado. No se realiza ninguna transformación de la geometría.
- Si no se proporciona como parámetro de entrada ningún identificador de sistemas de referencia espacial, se devuelve el identificador actual de la geometría especificada.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

Estas funciones se pueden invocar también como métodos.

### Sintaxis

```

>> [db2gse.ST_SrsId] ( [geometría] [ , -id_srs ] ) <<
    [db2gse.ST_SRID]
```

### Parámetros

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la que se define u obtiene el identificador de sistemas de referencia espacial.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial que se utilizará para la geometría resultante.

**Atención:** Si se especifica este parámetro, la geometría no se transforma, sino que se devuelve con su sistema de referencia espacial cambiado al sistema de referencia espacial especificado. Como resultado del cambio al nuevo sistema de referencia espacial, los datos pueden estar corrompidos. Para las transformaciones, utilice ST\_Transform.

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

## Tipos de retorno

- INTEGER, si no se especifica un *id\_srs*
- db2gse.ST\_Geometry, si se especifica un *id\_srs*

## Ejemplo

Este ejemplo crea dos puntos en dos sistemas de referencia espacial diferentes. El ID del sistema de referencia espacial correspondiente a cada punto se puede determinar utilizando la función ST\_SrsId.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points
VALUES (1, ST_Point( 'point (80 180)', 0 ) )
INSERT INTO sample_points
VALUES (2, ST_Point( 'point (-74.21450127 + 42.03415094)', 1 ) )

SELECT id, ST_SRSId (geometry) SRSID
FROM sample_points
```

Resultados:

ID	SRSID
1	0
2	1

---

## ST\_SrsName

ST\_SrsName toma una geometría como parámetro de entrada y devuelve el nombre del sistema de referencia espacial en el que está representada dicha geometría.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►►—db2gse.ST\_SrsName—(—geometría—)—————►►

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría para la que se devuelve el nombre del sistema de referencia espacial.

## Tipo de retorno

VARCHAR(128)

## Ejemplo

Este ejemplo crea dos puntos en sistemas de referencia espacial diferentes. Se utiliza la función ST\_SrsName para determinar el nombre del sistema de referencia espacial correspondiente a cada punto.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry, ST_Point)

INSERT INTO sample_points
VALUES (1, ST_Point ('point (80 180)', 0) )

INSERT INTO sample_points
VALUES (2, ST_Point ('point (-74.21450127 + 42.03415094)', 1) )

SELECT id, ST_SrsName (geometry) SRSNAME
FROM sample_points
```

Resultados:

ID	SRSNAME
1	DEFAULT_SRS
2	NAD83_SRS_1

---

## ST\_StartPoint

ST\_StartPoint toma una curva como parámetro de entrada y devuelve el primer punto de la curva. El punto resultante se representa según el sistema de referencia espacial de la curva proporcionada. Este resultado es equivalente al obtenido cuando se invoca a la función ST\_PointN(*curva*, 1)

Si la curva proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►►—db2gse.ST\_StartPoint—(—*curva*—)—————►►

## Parámetros

**curva** Valor de tipo ST\_Curve o uno de sus subtipos que representa la geometría a partir de la cual se obtiene el primer punto.

## Tipo de retorno

db2gse.ST\_Point

## Ejemplo

El ejemplo siguiente añade dos cadenas lineales a la tabla SAMPLE\_LINES. La primera cadena lineal tiene coordenadas X e Y. La segunda cadena lineal tiene coordenadas X, Y y Z. Se utiliza la función ST\_StartPoint para obtener el primer punto de cada cadena lineal.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)

INSERT INTO sample_lines
VALUES (1, ST_LineString ('linestring
(10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0))

INSERT INTO sample_lines
VALUES (1, ST_LineString ('linestring z
(0 0 4, 5 5 5, 10 10 6, 5 5 7, 0 0 8)', 0))

SELECT id, CAST( ST_AsText( ST_StartPoint( line ) ) AS VARCHAR(80))
START_POINT
FROM sample_lines
```

Resultados:

ID	START_POINT
1	POINT ( 10.00000000 10.00000000)
2	POINT Z ( 0.00000000 0.00000000 4.00000000)

---

## ST\_SymDifference

ST\_SymDifference toma como parámetros de entrada dos geometrías y devuelve la geometría que es la diferencia simétrica de las dos geometrías. La diferencia simétrica es la parte que no forma intersección de las dos geometrías proporcionadas. La geometría resultante se representa según el sistema de referencia espacial de la primera geometría. La dimensión de la geometría devuelta es la misma que la de las geometrías de entrada. Ambas geometrías deben tener la misma dirección.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

Si las geometrías son iguales, se devuelve una geometría vacía de tipo ST\_Point. Si cualquiera de las geometrías es un valor nulo, se devuelve el valor nulo.

La geometría resultante se representa según el tipo espacial más apropiado. Si se puede representar como punto, cadena lineal o polígono, se utiliza uno de estos tipos. En otro caso, se representa como multipunto, multilínea o multipolígono.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_SymDifference—(—*geometría1*—,—*geometría2*—)————►►

## Parámetros

### **geometría1**

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la primera geometría para calcular la diferencia simétrica con *geometría2*.

### **geometría2**

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la segunda geometría para calcular la diferencia simétrica con *geometría1*.

### **Restricciones para datos geodésicos:**

- Ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.
- ST\_SymDifference sólo soporta los tipos de datos ST\_Point, ST\_Polygon, ST\_MultiPoint y ST\_MultiPolygon.

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplos

### **Ejemplo 1**

En este ejemplo se muestra el uso de la función ST\_SymDifference. Las geometrías están almacenadas en la tabla SAMPLE\_GEOMS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms
VALUES( 1,
       ST_Geometry ('polygon ( (10 10, 10 20, 20 20, 20 10, 10 10) )', 0))

INSERT INTO sample_geoms
VALUES
(2, ST_Geometry ('polygon ( (30 30, 30 50, 50 50, 50 30, 30 30) )', 0))

INSERT INTO sample_geoms
VALUES
(3,ST_Geometry ('polygon ( (40 40, 40 60, 60 60, 60 40, 40 40) )', 0))

INSERT INTO sample_geoms
VALUES
(4, ST_Geometry ('linestring (70 70, 80 80)' , 0) )

INSERT INTO sample_geoms
VALUES
(5, ST_Geometry('linestring(75 75, 90 90)' ,0));
```

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. Los resultados variarán en función de la pantalla utilizada.

### **Ejemplo 2**

Este ejemplo utiliza ST\_SymDifference para obtener la diferencia simétrica de dos polígonos inconexos de la tabla SAMPLE\_GEOMS.

```

SELECT a.id, b.id,
       CAST (ST_AsText (ST_SymDifference (a.geometry, b.geometry) )
            AS VARCHAR(350) ) SYM_DIFF
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1 AND b.id = 2

```

Resultados:

```

ID  ID  SYM_DIFF
-----
1  2  MULTIPOLYGON ((( 10.00000000 10.00000000, 20.00000000 10.00000000,
                  20.00000000 20.00000000, 10.00000000 20.00000000,
                  10.00000000 10.00000000)),
              (( 30.00000000 30.00000000, 50.00000000 30.00000000,
                  50.00000000 50.00000000, 30.00000000 50.00000000,
                  30.00000000 30.00000000)))

```

### Ejemplo 3

Este ejemplo utiliza ST\_SymDifference para obtener la diferencia simétrica de dos polígonos que forman intersección de la tabla SAMPLE\_GEOMS.

```

SELECT a.id, b.id,
       CAST (ST_AsText (ST_SymDifference (a.geometry, b.geometry) )
            AS VARCHAR(500) ) SYM_DIFF
FROM sample_geoms a, sample_geoms b
WHERE a.id = 2 AND b.id = 3

```

Resultados:

```

ID  ID  SYM_DIFF
-----
2  3  MULTIPOLYGON ((( 40.00000000 50.00000000, 50.00000000 50.00000000,
                  50.00000000 40.00000000, 60.00000000 40.00000000,
                  60.00000000 60.00000000, 40.00000000 60.00000000,
                  40.00000000 50.00000000)),
              (( 30.00000000 30.00000000, 50.00000000 30.00000000,
                  50.00000000 40.00000000, 40.00000000 40.00000000,
                  40.00000000 50.00000000, 30.00000000 50.00000000,
                  30.00000000 30.00000000)))

```

### Ejemplo 4

Este ejemplo utiliza ST\_SymDifference para obtener la diferencia simétrica de dos cadenas lineales que forman intersección de la tabla SAMPLE\_GEOMS.

```

SELECT a.id, b.id,
       CAST (ST_AsText (ST_SymDifference (a.geometry, b.geometry) )
            AS VARCHAR(350) ) SYM_DIFF
FROM sample_geoms a, sample_geoms b
WHERE a.id = 4 AND b.id = 5

```

Resultados:

```

ID  ID  SYM_DIFF
-----
4  5  MULTILINESTRING (( 70.00000000 70.00000000, 75.00000000 75.00000000),
                  ( 80.00000000 80.00000000, 90.00000000 90.00000000))

```

---

## ST\_ToGeomColl

ST\_ToGeomColl toma una geometría como parámetro de entrada y la convierte a un conjunto de geometrías. El conjunto de geometrías resultante se representa según el sistema de referencia espacial de la geometría proporcionada.



Si la geometría especificada está vacía, puede ser de cualquier tipo. Pero se convierte a ST\_Multipoint, ST\_MultiLineString o ST\_MultiPolygon según convenga.

Si la geometría especificada no está vacía, debe ser de tipo ST\_Point, ST\_LineString o ST\_Polygon. Luego, la geometría se convierte al tipo ST\_Multipoint, ST\_MultiLineString o ST\_MultiPolygon respectivamente.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

```
►►—db2gse.ST_ToGeomColl—(—geometría—)—————►►
```

## Parámetro

### geometría

Valor de tipo ST\_Geometry o uno de los subtipos que representan la geometría que se convierte a un conjunto de geometrías.

## Tipo de retorno

db2gse.ST\_GeomCollection

## Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST\_ToGeomColl.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)
```

```
INSERT INTO sample_geometries
VALUES (1, ST_Polygon ('polygon ((3 3, 4 6, 5 3, 3 3))', 1)),
(2, ST_Point ('point (1 2)', 1))
```

La sentencia SELECT siguiente se utiliza la función ST\_ToGeomColl para obtener geometrías en forma de sus correspondientes subtipos de colección de geometrías.

```
SELECT id, CAST( ST_AsText( ST_ToGeomColl(geometry) )
AS VARCHAR(120) ) GEOM_COLL
FROM sample_geometries
```

Resultados:

```
ID          GEOM_COLL
-----
1 MULTIPOLYGON ((( 3.00000000 3.00000000, 5.00000000
                  3.00000000, 4.00000000 6.00000000,
                  3.00000000 3.00000000)))
2 MULTIPOINT ( 1.00000000 2.00000000)
```

---

## ST\_ToLineString

ST\_ToLineString toma una geometría como parámetro de entrada y la convierte en una cadena lineal. La cadena lineal resultante se representada según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser una cadena lineal. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

►►—db2gse.ST\_ToLineString—(*—geometría—*)—————►►

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se convierte en una cadena lineal.

Una geometría se puede convertir en una cadena lineal si está vacía o si es una cadena lineal. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

### Tipo de retorno

db2gse.ST\_LineString

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST\_ToLineString.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
VALUES (1, ST_Geometry ('linestring z (0 10 1, 0 0 3, 10 0 5)', 0)),
      (2, ST_Geometry ('point empty', 1) ),
      (3, ST_Geometry ('multipolygon empty', 1) )
```

La siguiente sentencia SELECT utiliza la función ST\_ToLineString para obtener cadenas lineales que se han convertido al tipo ST\_LineString a partir del tipo estático ST\_Geometry.

```
SELECT CAST( ST_AsText( ST_ToLineString(geometry) )
           AS VARCHAR(130) ) LINES
FROM sample_geometries
```

Resultados:

```
LINES
-----
LINESTRING Z ( 0.00000000 10.00000000 1.00000000, 0.00000000
```

```

          0.00000000 3.00000000, 10.00000000 0.00000000
          5.00000000)
LINESTRING EMPTY
LINESTRING EMPTY

```

---

## ST\_ToMultiLine

ST\_ToMultiLine toma una geometría como parámetro de entrada y la convierte en una multilínea. La multilínea resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser una multilínea o cadena lineal. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

► db2gse.ST\_ToMultiLine(—*geometría*—) ►

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se convierte en una multilínea.

Una geometría se puede convertir en una multilínea si está vacía, o si es una cadena lineal o multilínea. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

### Tipo de retorno

db2gse.ST\_MultiLineString

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST\_ToMultiLine.

```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
VALUES (1, ST_Geometry ('multilinestring ((0 10 1, 0 0 3, 10 0 5),
                                   (23 43, 27 34, 35 12))', 0) ),
       (2, ST_Geometry ('linestring z (0 10 1, 0 0 3, 10 0 5)', 0) ),
       (3, ST_Geometry ('point empty', 1) ),
       (4, ST_Geometry ('multipolygon empty', 1) )

```

La siguiente sentencia SELECT utiliza la función ST\_ToMultiLine para obtener multilíneas que se han convertido al tipo ST\_MultiLineString a partir del tipo estático ST\_Geometry.

```

SELECT CAST( ST_AsText( ST_ToMultiLine(geometry) )
AS VARCHAR(130) ) LINES
FROM sample_geometries

```

Resultados:

```
LINES
-----
MULTILINESTRING Z ( 0.00000000 10.00000000 1.00000000,
                    0.00000000 0.00000000 3.00000000,
                    10.00000000 0.00000000 5.00000000)
MULTILINESTRING EMPTY
MULTILINESTRING EMPTY
```

---

## ST\_ToMultiPoint

ST\_ToMultiPoint toma una geometría como parámetro de entrada y la convierte en multipunto. El multipunto resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser un punto o multipunto. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
►►—db2gse.ST_ToMultiPoint—(—geometría—)—————►
```

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se convierte en un multipunto.

Una geometría se puede convertir en un multipunto si está vacía, si es un punto o si es un multipunto. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

### Tipo de retorno

db2gse.ST\_MultiPoint

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST\_ToMultiPoint.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)
```

```
INSERT INTO sample_geometries
VALUES (1, ST_Geometry ('multipoint (0 0, 0 4)', 1) ),
       (2, ST_Geometry ('point (30 40)', 1) ),
       (3, ST_Geometry ('multipolygon empty', 1) )
```

La siguiente sentencia SELECT utiliza la función ST\_ToMultiPoint para obtener multipuntos que se han convertido al tipo ST\_MultiPoint a partir del tipo estático ST\_Geometry.

```
SELECT CAST( ST_AsText( ST_ToMultiPoint(geometry))
AS VARCHAR(62) ) MULTIPOINTS
FROM sample_geometries
```

Resultados:

```
MULTIPOINTS
-----
MULTIPOINT ( 0.00000000 0.00000000, 0.00000000 4.00000000)
MULTIPOINT ( 30.00000000 40.00000000)
MULTIPOINT EMPTY
```

---

## ST\_ToMultiPolygon

ST\_ToMultiPolygon toma una geometría como parámetro de entrada y la convierte en un multipolígono. El multipolígono resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser un polígono o multipolígono. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
►►—db2gse.ST_ToMultiPolygon—(—geometría—)—————►►
```

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se convierte en un multipolígono.

Una geometría se puede convertir en un multipolígono si está vacía, si es un polígono o un multipolígono. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

### Tipo de retorno

db2gse.ST\_MultiPolygon

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea varias geometrías y luego utiliza ST\_ToMultiPolygon para obtener multipolígonos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
VALUES (1, ST_Geometry ('polygon ((0 0, 0 4, 5 4, 5 0, 0 0))', 1)),
(2, ST_Geometry ('point empty', 1)),
(3, ST_Geometry ('multipoint empty', 1))
```

La siguiente sentencia SELECT utiliza la función ST\_ToMultiPolygon para obtener multipolígonos que se han convertido al tipo ST\_MultiPolygon a partir del tipo estático ST\_Geometry.

```
SELECT CAST( ST_AsText( ST_ToMultiPolygon(geometry) )
AS VARCHAR(130) ) POLYGONS
FROM sample_geometries
```

Resultados:

POLYGONS

```
-----
MULTIPOLYGON (( 0.00000000 0.00000000, 5.00000000 0.00000000,
                 5.00000000 4.00000000, 0.00000000 4.00000000,
                 0.00000000 0.00000000))
```

MULTIPOLYGON EMPTY

MULTIPOLYGON EMPTY

---

## ST\_ToPoint

ST\_ToPoint toma una geometría como parámetro de entrada y la convierte en un punto. El punto resultante se representa utilizando el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser un punto. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
►►—db2gse.ST_ToPoint—(—geometría—)—————►►
```

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría que se convierte a punto.

Una geometría se puede convertir a un punto si está vacía o es un punto. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

### Tipo de retorno

db2gse.ST\_Point

### Ejemplo

Este ejemplo crea tres geometrías en SAMPLE\_GEOMETRIES y convierte cada geometría en un punto.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
```

```
VALUES (1, ST_Geometry ('point (30 40)', 1) ),
       (2, ST_Geometry ('linestring empty', 1) ),
       (3, ST_Geometry ('multipolygon empty', 1) )
```

La siguiente sentencia SELECT utiliza la función ST\_ToPoint para obtener puntos que se han convertido al tipo ST\_Point a partir del tipo estático ST\_Geometry.

```
SELECT CAST( ST_AsText( ST_ToPoint(geometry) ) AS VARCHAR(35) ) POINTS
FROM sample_geometries
```

Resultados:

```
POINTS
-----
POINT ( 30.00000000 40.00000000)
POINT EMPTY
POINT EMPTY
```

---

## ST\_ToPolygon

ST\_ToPolygon toma una geometría como parámetro de entrada y la convierte en un polígono. El polígono resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser un polígono. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
►►—db2gse.ST_ToPolygon—(—geometría—)—————►
```

### Parámetro

#### geometría

Valor de tipo ST\_Geometry o uno de sus subtipos que representa la geometría que se convierte a un polígono.

Una geometría se puede convertir a un polígono si está vacía o es un polígono. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

### Tipo de retorno

db2gse.ST\_Polygon

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea tres geometrías en SAMPLE\_GEOMETRIES y convierte cada geometría en un polígono.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)
```

```

INSERT INTO sample_geometries
VALUES (1, ST_Geometry ('polygon ((0 0, 0 4, 5 4, 5 0, 0 0))', 1) ),
       (2, ST_Geometry ('point empty', 1) ),
       (3, ST_Geometry ('multipolygon empty', 1) )

```

La siguiente sentencia SELECT utiliza la función ST\_ToPolygon para obtener polígonos que se han convertido al tipo ST\_Polygon a partir del tipo estático ST\_Geometry.

```

SELECT CAST( ST_AsText( ST_ToPolygon(geometry) ) AS VARCHAR(130) ) POLYGONS
FROM sample_geometries

```

Resultados:

POLYGONS

```

-----
POLYGON (( 0.00000000 0.00000000, 5.00000000 0.00000000,
           5.00000000 4.00000000,0.00000000 4.00000000,
           0.00000000 0.00000000))

```

POLYGON EMPTY

POLYGON EMPTY

---

## ST\_Touches

ST\_Touches toma dos geometrías como parámetros de entrada y devuelve un 1 si las geometrías proporcionadas están espacialmente en contacto. En caso contrario, se devuelve un 0 (cero).

Dos geometrías están en contacto si los interiores de ambas geometrías no forman intersección, pero el perímetro de una de ellas forma intersección con el perímetro o el interior de la otra geometría.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si ambas geometrías proporcionadas son puntos o multipuntos, o si cualquiera de las dos geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

### Sintaxis

```

▶▶—db2gse.ST_Touches—(—geometría1—,—geometría2—)—▶▶

```

### Parámetros

#### geometría1

Un valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría que se debe verificar si está en contacto con *geometría2*.

#### geometría2

Un valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría que se debe verificar si está en contacto con *geometría1*.

### Tipo de retorno

INTEGER



## Ejemplo

Este ejemplo añade varias geometrías a la tabla SAMPLE\_GEOMS. Luego, se utiliza la función ST\_Touches para determinar qué geometrías están en contacto entre sí.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms
VALUES (1, ST_Geometry('polygon ( (20 30, 30 30, 30 40, 20 40, 20 30) )' , 0) )

INSERT INTO sample_geoms
VALUES (2, ST_Geometry('polygon ( (30 30, 30 50, 50 50, 50 30, 30 30) )' ,0) )

INSERT INTO sample_geoms
VALUES (3, ST_Geometry('polygon ( (40 40, 40 60, 60 60, 60 40, 40 40) )' , 0) )

INSERT INTO sample_geoms
VALUES (4, ST_Geometry('linestring( 60 60, 70 70 )' , 0) )

INSERT INTO sample_geoms
VALUES (5, ST_Geometry('linestring( 30 30, 60 60 )' , 0) )

SELECT a.id, b.id, ST_Touches (a.geometry, b.geometry) TOUCHES
FROM sample_geoms a, sample_geoms b
WHERE b.id >= a.id
```

Resultados:

ID	ID	TOUCHES
1	1	0
1	2	1
1	3	0
1	4	0
1	5	1
2	2	0
2	3	0
2	4	0
2	5	1
3	3	0
3	4	1
3	5	1
4	4	0
4	5	1
5	5	0

---

## ST\_Transform

ST\_Transform toma como parámetros de entrada una geometría y un identificador de sistemas de referencia espacial y transforma la geometría para que esté representada en el sistema de referencia espacial especificado. Se realizan proyecciones y conversiones entre diferentes sistemas de coordenadas y se ajustan las coordenadas de las geometrías según convenga.

La geometría se puede convertir al sistema de referencia espacial especificado sólo si éste está basado en el mismo sistema de coordenadas geográficas que el sistema de referencia espacial actual de la geometría. Si el sistema de referencia espacial especificado o el utilizado actualmente por la geometría está basado en un sistema de coordenadas proyectadas, se realiza una proyección inversa para determinar el sistema de coordenadas geográficas en el que se basa el sistema proyectado.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

►►—db2gse.ST\_Transform—(—*geometría*—,—*id\_srs*—)—————►►

## Parámetros

### **geometría**

Un valor de tipo ST\_Geometry o uno de los subtipos que representa la geometría que se transforma en el sistema de referencia espacial identificado por *id\_srs*.

**id\_srs** Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

Si la transformación en el sistema de referencia espacial especificado no se puede llevar a cabo porque el sistema de referencia espacial actual *geometría* no es compatible con el sistema de referencia espacial identificado por *id\_srs*, se emite una condición de excepción (SQLSTATE 38SUC).

Si *id\_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST\_SPATIAL\_REFERENCE\_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplos

### Ejemplo 1

En el ejemplo siguiente muestra el uso de ST\_Transform para convertir una geometría desde un sistema de referencia espacial a otro.

Primero, se invoca a db2se para crear un sistema de referencia espacial de plano de estado cuyo ID sea 3.

```
db2se create_srs SAMP_DB
-srsId 3 -srsName z3101a -xOffset 0 -yOffset 0 -xScale 1 -yScale 1
- coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet
```

A continuación se añaden puntos a:

- La tabla SAMPLE\_POINTS\_SP con coordenadas de plano de estado utilizando ese sistema de referencia espacial.
- la tabla SAMPLE\_POINTS\_LL utilizando coordenadas especificadas de latitud y longitud.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points_sp (id INTEGER, geometry ST_Point)
CREATE TABLE sample_points_ll (id INTEGER, geometry ST_Point)

INSERT INTO sample_points_sp
VALUES (12457, ST_Point('point ( 567176.0 1166411.0)', 3) )

INSERT INTO sample_points_sp
VALUES (12477, ST_Point('point ( 637948.0 1177640.0)', 3) )
```

```

INSERT INTO sample_points_ll
VALUES (12457, ST_Point('point ( -74.22371600 42.03498700)', 1) )

INSERT INTO sample_points_ll
VALUES (12477, ST_Point('point ( -73.96293200 42.06487900)', 1) )

```

Luego, se utiliza la función ST\_Transform para convertir las geometrías.

### Ejemplo 2

Este ejemplo convierte puntos con coordenadas de latitud y longitud en coordenadas de plano de estado.

```

SELECT id, CAST( ST_AsText( ST_Transform( geometry, 3) )
AS VARCHAR(100) ) STATE_PLANE
FROM sample_points_ll

```

Resultados:

ID	STATE_PLANE
12457	POINT ( 567176.00000000 1166411.00000000)
12477	POINT ( 637948.00000000 1177640.00000000)

### Ejemplo 3

Este ejemplo convierte puntos con coordenadas de plano de estado en coordenadas de latitud y longitud.

```

SELECT id, CAST( ST_AsText( ST_Transform( geometry, 1) )
AS VARCHAR(100) ) LAT_LONG
FROM sample_points_sp

```

Resultados:

ID	LAT_LONG
12457	POINT ( -74.22371500 42.03498800)
12477	POINT ( -73.96293100 42.06488000)

---

## ST\_Union

ST\_Union toma dos geometrías como parámetros de entrada y devuelve la geometría que es la unión de las geometrías proporcionadas. La geometría resultante se representa según el sistema de referencia espacial de la primera geometría.

Ambas geometrías deben tener la misma dirección. Si cualquiera de las dos geometrías proporcionadas es nulo, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

La geometría resultante se representa según el tipo espacial más apropiado. Si se puede representar como punto, cadena lineal o polígono, se utiliza uno de estos tipos. En otro caso, se representa como multipunto, multilínea o multipolígono.

También se puede invocar esta función como método.

## Sintaxis

► db2gse.ST\_Union(—*geometría1*—,—*geometría2*—) ◀

## Parámetros

### *geometría1*

Valor de tipo ST\_Geometry o uno de sus subtipos que se combina con *geometría2*.

### *geometría2*

Valor de tipo ST\_Geometry o uno de sus subtipos que se combina con *geometría1*.

**Restricciones para los datos geodésicos:** Ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

## Tipo de retorno

db2gse.ST\_Geometry

## Ejemplos

### Ejemplo 1

Las siguientes sentencias de SQL crean y llenan la tabla SAMPLE\_GEOMS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry, ST_Geometry)
```

```
INSERT INTO sample_geoms
VALUES (1, ST_Geometry( 'polygon
((10 10, 10 20, 20 20, 20 10, 10 10) )', 0))
```

```
INSERT INTO sample_geoms
VALUES (2, ST_Geometry( 'polygon
((30 30, 30 50, 50 50, 50 30, 30 30) )', 0))
```

```
INSERT INTO sample_geoms
VALUES (3, ST_Geometry( 'polygon
((40 40, 40 60, 60 60, 60 40, 40 40) )', 0))
```

```
INSERT INTO sample_geoms
VALUES (4, ST_Geometry('linestring (70 70, 80 80)', 0))
```

```
INSERT INTO sample_geoms
VALUES (5, ST_Geometry('linestring (80 80, 100 70)', 0))
```

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. Los resultados dependerán de la pantalla utilizada.

### Ejemplo 2

Este ejemplo obtiene la unión de dos polígonos inconexos.

```
SELECT a.id, b.id, CAST ( ST_AsText( ST_Union( a.geometry, b.geometry) )
AS VARCHAR (350) ) UNION
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1 AND b.id = 2
```

Resultados:

```
ID    ID    UNION
-----
1      2 MULTIPOLYGON ((( 10.00000000 10.00000000, 20.00000000
10.00000000, 20.00000000 20.00000000, 10.00000000
20.00000000, 10.00000000 10.00000000))
(( 30.00000000 30.00000000, 50.00000000
30.00000000,50.00000000 50.00000000, 30.00000000
50.00000000,30.00000000 30.00000000)))
```

### Ejemplo 3

Este ejemplo obtiene la unión de dos polígonos que forman intersección.

```
SELECT a.id, b.id, CAST ( ST_AsText( ST_Union(a.geometry, b.geometry))
AS VARCHAR (250)) UNION
FROM sample_geoms a, sample_geoms b
WHERE a.id = 2 AND b.id = 3
```

Resultados:

```
ID    ID    UNION
-----
2      3 POLYGON (( 30.00000000 30.00000000, 50.00000000
30.00000000,50.00000000 40.00000000, 60.00000000
40.00000000,60.00000000 60.00000000, 40.00000000
60.00000000 40.00000000 50.00000000, 30.00000000
50.00000000, 30.00000000 30.00000000))
```

### Ejemplo 4

Este ejemplo obtiene la unión de dos cadenas lineales.

```
SELECT a.id, b.id, CAST ( ST_AsText( ST_Union( a.geometry, b.geometry) )
AS VARCHAR (250) ) UNION
FROM sample_geoms a, sample_geoms b
WHERE a.id = 4 AND b.id = 5
```

Resultados:

```
ID    ID    UNION
-----
4      5 MULTILINESTRING (( 70.00000000 70.00000000, 80.00000000 80.00000000),
( 80.00000000 80.00000000, 100.00000000 70.00000000))
```

---

## ST\_Within

ST\_Within toma dos geometrías como parámetros de entrada y devuelve un 1 si la primera geometría está completamente dentro de la segunda. En caso contrario, se devuelve un 0 (cero).

Si cualquiera de las geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

ST\_Within efectúa la misma operación lógica que ST\_Contains, con los parámetros invertidos.

## Sintaxis

►—db2gse.ST\_Within—(—geometría1—,—geometría2—)—◄

## Parámetros

### geometría1

Un valor de tipo ST\_Geometry o uno de los subtipos para el que se debe comprobar si está completamente dentro de *geometría2*.

### geometría2

Un valor de tipo ST\_Geometry o uno de los subtipos que se debe comprobar si está completamente dentro de *geometría1*.

**Restricciones para los datos geodésicos:** Ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

## Tipo de retorno

INTEGER

## Ejemplos

### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_Within. Se crean e insertan geometrías en tres tablas: SAMPLE\_POINTS, SAMPLE\_LINES y SAMPLE\_POLYGONS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)
CREATE TABLE sample_polygons (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_points (id, geometry)
VALUES (1, ST_Point (10, 20, 1) ),
       (2, ST_Point ('point (41 41)', 1) )

INSERT INTO sample_lines (id, line)
VALUES (10, ST_LineString ('linestring (1 10, 3 12, 10 10)', 1) ),
       (20, ST_LineString ('linestring (50 10, 50 12, 45 10)', 1) )

INSERT INTO sample_polygons (id, geometry)
VALUES (100, ST_Polygon ('polygon (( 0 0, 0 40, 40 40, 40 0, 0 0))', 1) )
```

### Ejemplo 2

Este ejemplo obtiene puntos de la tabla SAMPLE\_POINTS que están en polígonos de la tabla SAMPLE\_POLYGONS.

```
SELECT a.id POINT_ID_WITHIN_POLYGONS
FROM sample_points a, sample_polygons b
WHERE ST_Within( b.geometry, a.geometry) = 0
```

Resultados:

```
POINT_ID_WITHIN_POLYGONS
-----
2
```

### Ejemplo 3

Este ejemplo obtiene cadenas lineales de la tabla SAMPLE\_LINES que están en polígonos de la tabla SAMPLE\_POLYGONS.

```
SELECT a.id LINE_ID_WITHIN_POLYGONS
FROM sample_lines a, sample_polygons b
WHERE ST_Within( b.geometry, a.geometry) = 0
```

Resultados:

```
LINE_ID_WITHIN_POLYGONS
-----
1
```

---

## ST\_WKBTToSQL

ST\_WKBTToSQL toma una representación binaria convencional de una geometría y devuelve la geometría correspondiente. El sistema de referencia espacial con el identificador 0 (cero) se utiliza para la geometría resultante.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

ST\_WKBTToSQL(*wkb*) produce el mismo resultado que ST\_Geometry(*wkb*,0). Debido a su flexibilidad, es recomendable utilizar la función ST\_Geometry en lugar de ST\_WKBTToSQL: ST\_Geometry utiliza otras modalidades de datos de entrada además de la representación binaria convencional.

### Sintaxis

►► db2gse.ST\_WKBTToSQL(—*wkb*—) ◀◀

### Parámetro

**wkb** Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la geometría resultante.

### Tipo de retorno

db2gse.ST\_Geometry

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST\_WKBTToSQL. Primero, se almacenan geometrías en la columna GEOMETRY de la tabla SAMPLE\_GEOMETRIES. Luego, sus representaciones binarias convencionales se guardan en la columna WKB utilizando la función ST\_AsBinary en la sentencia UPDATE. Finalmente, se utiliza la función ST\_WKBTToSQL para obtener las coordenadas de las geometrías contenidas en la columna WKB.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries
(id INTEGER, geometry ST_Geometry, wkb BLOB(32K) )
```

```
INSERT INTO sample_geometries (id, geometry)
VALUES (10, ST_Point ( 'point (44 14)', 0 ) ),
```

```

        (11, ST_Point ( 'point (24 13)', 0 ) ),
        (12, ST_Polygon ('polygon ((50 20, 50 40, 70 30, 50 20))', 0 ) )
UPDATE sample_geometries AS temp_correlated
SET wkb = ST_AsBinary(geometry)
WHERE id = temp_correlated.id

```

Utilice esta sentencia SELECT para ver las geometrías contenidas en la columna WKB.

```

SELECT id, CAST( ST_AsText( ST_WKBToSQL(wkb) ) AS VARCHAR(120) ) GEOMETRIES
FROM sample_geometries

```

Resultados:

ID	GEOMETRIES
10	POINT ( 44.00000000 14.00000000)
11	POINT ( 24.00000000 13.00000000)
12	POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000, 50.00000000 40.00000000, 50.00000000 20.00000000))

---

## ST\_WKTTToSQL

ST\_WKTTToSQL toma una representación de texto convencional de una geometría y devuelve la geometría correspondiente. El sistema de referencia espacial con el identificador 0 (cero) se utiliza para la geometría resultante.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

ST\_WKTTToSQL(*wkt*) produce el mismo resultado que ST\_Geometry(*wkt*,0). Debido a su flexibilidad, es recomendable utilizar la función ST\_Geometry en lugar de ST\_WKTTToSQL: ST\_Geometry utiliza otras modalidades de datos de entrada además de la representación de texto convencional.

### Sintaxis

►►—db2gse.ST\_WKTTToSQL—(—*wkt*—)—————►►

### Parámetro

**wkt** Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la geometría resultante.

### Tipo de retorno

db2gse.ST\_Geometry

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de ST\_WKTTToSQL para crear e insertar geometrías utilizando sus representaciones de texto convencionales.



```

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
VALUES (10, ST_WKTToSQL( 'point (44 14)' ) ),
      (11, ST_WKTToSQL ( 'point (24 13)' ) ),
      (12, ST_WKTToSQL ('polygon ((50 20, 50 40, 70 30, 50 20))' ) )

```

Esta sentencia SELECT obtiene las geometrías que se han insertado.

```

SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(120) ) GEOMETRIES
FROM sample_geometries

```

Resultados:

ID	GEOMETRIES
10	POINT ( 44.00000000 14.00000000)
11	POINT ( 24.00000000 13.00000000)
12	POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000, 50.00000000 40.00000000, 50.00000000 20.00000000))

---

## ST\_X

ST\_X puede actuar de una de estas maneras:

- Toma un punto como parámetro de entrada y devuelve su coordenada X
- Toma un punto y una coordenada X como parámetros de entrada y devuelve el propio punto con su coordenada X establecida en el valor proporcionado.

Si el punto proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```

▶▶ db2gse.ST_X ( ( punto [ , coordenada_x ] ) ) ▶▶

```

### Parámetros

**punto** Valor de tipo ST\_Point para el cual se obtiene o modifica la coordenada X.

**coordenada\_x**

Valor de tipo DOUBLE que representa la nueva coordenada X del *punto*.

### Tipos de retorno

- DOUBLE, si no se especifica *coordenada\_x*
- db2gse.ST\_Point, si se especifica *coordenada\_x*

### Ejemplos

#### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_X. Se crean e insertan geometrías en la tabla SAMPLE\_POINTS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
```

```
INSERT INTO sample_points (id, geometry)
VALUES (1, ST_Point (2, 3, 32, 5, 1) ),
       (2, ST_Point (4, 5, 20, 4, 1) ),
       (3, ST_Point (3, 8, 23, 7, 1) )
```

### Ejemplo 2

Este ejemplo determina las coordenadas X de los puntos contenidos en la tabla.

```
SELECT id, ST_X (geometry) X_COORD
FROM sample_points
```

Resultados:

ID	X_COORD
1	+2.00000000000000E+000
2	+4.00000000000000E+000
3	+3.00000000000000E+000

### Ejemplo 3

Este ejemplo devuelve un punto con su coordenada X establecida en el valor 40.

```
SELECT id, CAST( ST_AsText( ST_X (geometry, 40)) AS VARCHAR(60) )
X_40
FROM sample_points
WHERE id=3
```

Resultados:

ID	X_40
3	POINT ZM ( 40.00000000 8.00000000 23.00000000 7.00000000)

---

## ST\_Y

ST\_Y puede actuar de una de estas maneras:

- Toma un punto como parámetro de entrada y devuelve su coordenada Y
- Toma un punto y una coordenada Y como parámetros de entrada y devuelve el propio punto con su coordenada Y establecida en el valor proporcionado

Si el punto proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar esta función como método.

### Sintaxis

```
db2gse.ST_Y(—punto—, —coordenada_y—)
```

### Parámetros

**punto** Valor de tipo ST\_Point para el cual se obtiene o modifica la coordenada Y.

**coordenada\_y**

Valor de tipo DOUBLE que representa la nueva coordenada Y del *punto*.

## Tipos de retorno

- DOUBLE, si no se especifica *coordenada\_y*
- db2gse.ST\_Point, si se especifica *coordenada\_y*

## Ejemplos

### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_Y. Se crean e insertan geometrías en la tabla SAMPLE\_POINTS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
```

```
INSERT INTO sample_points (id, geometry)
VALUES (1, ST_Point (2, 3, 32, 5, 1) ),
       (2, ST_Point (4, 5, 20, 4, 1) ),
       (3, ST_Point (3, 8, 23, 7, 1) )
```

### Ejemplo 2

Este ejemplo determina las coordenadas Y de los puntos contenidos en la tabla.

```
SELECT id, ST_Y (geometry) Y_COORD
FROM sample_points
```

Resultados:

ID	Y_COORD
1	+3.0000000000000000E+000
2	+5.0000000000000000E+000
3	+8.0000000000000000E+000

### Ejemplo 3

Este ejemplo devuelve un punto con su coordenada Y establecida en el valor 40.

```
SELECT id, CAST( ST_AsText( ST_Y (geometry, 40)) AS VARCHAR(60) )
Y_40
FROM sample_points
WHERE id=3
```

Resultados:

ID	Y_40
3	POINT ZM ( 3.00000000 40.00000000 23.00000000 7.00000000)

---

## ST\_Z

ST\_Z puede actuar de una de estas maneras:

- Toma un punto como parámetro de entrada y devuelve su coordenada Z
- Toma como parámetros de entrada un punto y una coordenada Z y devuelve el propio punto junto con su coordenada Z establecida en el valor proporcionado, aunque el punto especificado no tenga ninguna coordenada Z.

Si la coordenada Z especificada es nula, la coordenada Z se elimina del punto.

Si el punto especificado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar esta función como método.

## Sintaxis

```
db2gse.ST_Z(punto, coordenada_z)
```

## Parámetros

**punto** Valor de tipo ST\_Point para el cual se obtiene o modifica la coordenada Z.

**coordenada\_z**

Valor de tipo DOUBLE que representa la nueva coordenada Z del *punto*.

Si *coordenada\_z* es nulo, la coordenada Z se elimina de *punto*.

## Tipos de retorno

- DOUBLE, si no se especifica *coordenada\_z*
- db2gse.ST\_Point, si se especifica *coordenada\_z*

## Ejemplos

### Ejemplo 1

En este ejemplo se muestra el uso de la función ST\_Z. Se crean e insertan geometrías en la tabla SAMPLE\_POINTS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
```

```
INSERT INTO sample_points (id, geometry)
VALUES (1, ST_Point (2, 3, 32, 5, 1) ),
       (2, ST_Point (4, 5, 20, 4, 1) ),
       (3, ST_Point (3, 8, 23, 7, 1) )
```

### Ejemplo 2

Este ejemplo determina las coordenadas Z de los puntos contenidos en la tabla.

```
SELECT id, ST_Z (geometry) Z_COORD
FROM sample_points
```

Resultados:

ID	Z_COORD
1	+3.20000000000000E+001
2	+2.00000000000000E+001
3	+2.30000000000000E+001

### Ejemplo 3

Este ejemplo devuelve un punto con su coordenada Z establecida en el valor 40.

```
SELECT id, CAST( ST_AsText( ST_Z (geometry, 40)) AS VARCHAR(60) )
       Z_40
FROM sample_points
WHERE id=3
```

Resultados:

---

## Union aggregate (Agregado de unión)

Un agregado de unión es la combinación de las funciones `ST_BuildUnionAggr` y `ST_GetAggrResult`. Esta combinación agrega una columna de geometrías de una tabla a una geometría individual mediante la creación de la unión.

Si todas las geometrías que se deben combinar en la unión son nulas, se devuelve un valor nulo. Si cualquiera de las geometrías que se deben combinar en la unión son nulas o están vacías, se devuelve una geometría vacía de tipo `ST_Point`.

La función `ST_BuildUnionAggr` también se puede invocar como método.

### Sintaxis

```
db2gse.ST_GetAggrResult(----->
MAX(---db2gse.ST_BuildUnionAggr(---geometrías---)---)>>
```

### Parámetros

#### geometrías

Columna de una tabla cuyo tipo es `ST_Geometry` o uno de sus subtipos que representa todas las geometrías que se deben combinar en una unión.

### Tipo de retorno

`db2gse.ST_Geometry`

### Restricciones

No puede crear el agregado de unión de una columna espacial de una tabla en los casos siguientes:

- En entornos de Database Partitioning Feature (DPF)
- Si se utiliza una cláusula `GROUP BY` en la sentencia `SELECT`
- Si utiliza una función distinta de la función de agregación `MAX` de DB2

### Ejemplo

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de un agregado de unión para combinar un conjunto de puntos y formar multipuntos. Primero se añaden varios puntos a la tabla `SAMPLE_POINTS`. Luego se utilizan las funciones `ST_GetAggrResult` y `ST_BuildUnionAggr` para crear la unión de los puntos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
```

```
INSERT INTO sample_points
VALUES (1, ST_Point (2, 3, 1) )
```

```

INSERT INTO sample_points
VALUES (2, ST_Point (4, 5, 1) )
INSERT INTO sample_points
VALUES (3, ST_Point (13, 15, 1) )
INSERT INTO sample_points
VALUES (4, ST_Point (12, 5, 1) )
INSERT INTO sample_points
VALUES (5, ST_Point (23, 2, 1) )
INSERT INTO sample_points
VALUES (6, ST_Point (11, 4, 1) )

SELECT CAST (ST_AsText(
    ST_GetAggrResult( MAX( ST_BuildUnionAggregate (geometry) ) ))
AS VARCHAR(160)) POINT_AGGREGATE
FROM sample_points

```

Resultados:

POINT\_AGGREGATE

```

-----
MULTIPOINT ( 2.00000000 3.00000000, 4.00000000 5.00000000,
             11.00000000 4.00000000, 12.00000000 5.00000000,
             13.00000000 15.00000000, 23.00000000 2.00000000)

```

---

## Capítulo 24. Grupos de transformación

---

### Grupos de transformación

Spatial Extender proporciona cuatro grupos de transformación que se utilizan para transferir geometrías entre el servidor de DB2 y una aplicación cliente. Estos grupos de transformación incluyen los siguientes formatos de intercambio de datos:

- Representación de texto convencional (well-known text, WKT)
- Representación binaria convencional (well-known binary, WKB)
- Representación de forma ESRI
- Geography Markup Language (GML)

Cuando se recuperan datos de una tabla que contiene una columna espacial, los datos de la columna espacial se transforman en un valor de tipo CLOB(2G) o BLOB(2G), según si se ha indicado que los datos transformados se deben representar en formato binario o de texto. También puede utilizar los grupos de transformación para transferir datos espaciales a la base de datos.

Para seleccionar qué grupo de transformación se debe utilizar al transferir datos, utilice la sentencia `SET CURRENT DEFAULT TRANSFORM GROUP` para modificar el registro especial `CURRENT DEFAULT TRANSFORM GROUP` de DB2. DB2 utiliza el valor de este registro especial para determinar qué funciones de transformación se deben invocar para realizar las conversiones necesarias.

Los grupos de transformación pueden simplificar la programación de aplicaciones. En lugar de utilizar explícitamente funciones de conversión en las sentencias de SQL, puede especificar un grupo de transformación y dejar que DB2 se ocupe de la tarea.

---

### Grupo de transformación `ST_WellKnownText`

Puede utilizar el grupo de transformación `ST_WellKnownText` para intercambiar datos con DB2<sup>®</sup> utilizando la representación de texto convencional (WKT).

Cuando se vincula un valor del servidor de bases de datos con el cliente, se utiliza la misma función proporcionada por `ST_AsText()` para convertir una geometría a la representación WKT. Cuando se transfiere la representación de texto convencional de una geometría al servidor de bases de datos, se utiliza implícitamente la función `ST_Geometry(CLOB)` para realizar las conversiones a un valor de tipo `ST_Geometry`. El uso del grupo de transformación para vincular valores con DB2 hace que las geometrías se representen en el sistema de referencia espacial con el identificador numérico 0 (cero).

#### Ejemplos

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

#### Ejemplo 1

El siguiente script de SQL muestra cómo utilizar el grupo de transformación ST\_WellKnownText para obtener una geometría en su representación de texto convencional sin utilizar la función explícita ST\_AsText.

```
CREATE TABLE transforms_sample (
    id INTEGER,
    geom db2gse.ST_Geometry)

INSERT
  INTO transforms_sample
  VALUES (1, db2gse.ST_LineString('linestring
    (100 100, 200 100)', 0))

SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownText

SELECT id, geom
  FROM transforms_sample
  WHERE id = 1
```

Resultados:

```
ID      GEOM
-----
1  LINESTRING ( 100.000000000 100.000000000, 200.000000000 100.000000000)
```

## Ejemplo 2

El siguiente código en C muestra cómo utilizar el grupo de transformación ST\_WellKnownText para insertar geometrías utilizando la función explícita ST\_Geometry para la variable de lenguaje principal wkt\_buffer, cuyo tipo es CLOB y que contiene la representación de texto convencional del punto (10 10) que se debe insertar.

```
EXEC SQL BEGIN DECLARE SECTION;
    sqlint32 id = 0;
    SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) wkt_buffer;
EXEC SQL END DECLARE SECTION;

// definir el grupo de transformación para todas las sentencias de SQL
// posteriores
EXEC SQL
    SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownText;

id = 100;
strcpy(wkt_buffer.data, "point ( 10 10 )");
wkt_buffer.length = strlen(wkt_buffer.data);

// insertar punto con WKT en la columna de tipo ST_Geometry
EXEC SQL
    INSERT
    INTO transforms_sample(id, geom)
    VALUES (:id, :wkt_buffer);
```

---

## Grupo de transformación ST\_WellKnownBinary

Utilice el grupo de transformación ST\_WellKnownBinary para intercambiar datos con DB2® utilizando la representación de texto convencional (WKB).

Cuando se vincula un valor del servidor de bases de datos con el cliente, se utiliza la misma función proporcionada por ST\_AsBinary() para convertir una geometría a la representación WKB. Cuando se transfiere la representación binaria convencional de una geometría al servidor de bases de datos, se utiliza implícitamente la función ST\_Geometry(BLOB) para realizar las conversiones a un valor de tipo



ST\_Geometry. El uso del grupo de transformación para vincular valores con DB2 hace que las geometrías se representen en el sistema de referencia espacial con el identificador numérico 0 (cero).

## Ejemplo

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

### Ejemplo 1

El siguiente script de SQL muestra cómo utilizar el grupo de transformación ST\_WellKnownBinary para obtener una geometría en su representación binaria convencional sin utilizar la función explícita ST\_AsBinary.

```
CREATE TABLE transforms_sample (
  id INTEGER,
  geom db2gse.ST_Geometry)

INSERT
  INTO transforms_sample
  VALUES ( 1, db2gse.ST_Polygon('polygon ((10 10, 20 10, 20 20,
    10 20, 10 10))', 0))

SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary

SELECT id, geom
  FROM transforms_sample
  WHERE id = 1
```

Resultados:

```
ID      GEOM
-----
1 x'010300000001000000050000000000000000000000244000
  0000000000244000000000000000244000000000000003440
  000000000000344000000000000000344000000000000034
  4000000000000024400000000000002440000000000000
  2440'
```

### Ejemplo 2

El siguiente código en C muestra cómo utilizar el grupo de transformación ST\_WellKnownBinary para insertar geometrías utilizando la función explícita ST\_Geometry para la variable de lenguaje principal wkb\_buffer, cuyo tipo es BLOB y que contiene la representación binaria convencional de una geometría que se debe insertar.

```
EXEC SQL BEGIN DECLARE SECTION;
  sqlint32 id = 0;
  SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) wkb_buffer;
EXEC SQL END DECLARE SECTION;

// definir el grupo de transformación para todas las sentencias de SQL
// posteriores
EXEC SQL
  SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary;

// inicializar variables de lenguaje principal
...
// insertar geometría con WKB en la columna de tipo ST_Geometry
```

```
EXEC SQL
  INSERT
    INTO transforms_sample(id, geom)
    VALUES ( :id, :wkb_buffer );
```

---

## Grupo de transformación ST\_Shape

Utilice el grupo ST\_Shape para intercambiar datos con DB2® utilizando la representación de formas ESRI.

Cuando se vincula un valor del servidor de bases de datos con el cliente, se utiliza la misma función proporcionada por ST\_AsShape() para convertir una geometría en su representación de forma. Cuando se transfiere la representación de forma de una geometría al servidor de bases de datos, se utiliza implícitamente la función ST\_Geometry(BLOB) para realizar las conversiones a un valor de tipo ST\_Geometry. El uso del grupo de transformación para vincular valores con DB2 hace que las geometrías se representen en el sistema de referencia espacial con el identificador numérico 0 (cero).

### Ejemplos

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

#### Ejemplo 1

El siguiente script de SQL muestra el uso del grupo de transformación ST\_Shape para obtener una geometría en su representación forma, sin utilizar la función explícita ST\_AsShape.

```
CREATE TABLE transforms_sample(
  id INTEGER,
  geom db2gse.ST_Geometry)

INSERT
  INTO transforms_sample
  VALUES ( 1, db2gse.ST_Point(20.0, 30.0, 0) )

SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape

SELECT id, geom
  FROM transforms_sample
 WHERE id = 1
```

Resultados:

```
ID      GEOM
-----
 1 x'01000000000000000000000000000000344000000000000003E40'
```

#### Ejemplo 2

El siguiente código en C muestra cómo utilizar el grupo de transformación ST\_Shape para insertar geometrías utilizando la función explícita ST\_Geometry para la variable de lenguaje principal shape\_buffer, cuyo tipo es BLOB y que contiene la representación de forma de una geometría que se debe insertar.

```
EXEC SQL BEGIN DECLARE SECTION;
  sqlint32 id = 0;
  SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) shape_buffer;
EXEC SQL END DECLARE SECTION;
```

```

// definir el grupo de transformación para todas las sentencias de SQL
// posteriores
EXEC SQL
    SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// inicializar variables de lenguaje principal
...

SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// insertar geometría con representación de forma en la columna de tipo ST_Geometry
EXEC SQL
    INSERT
    INTO transforms_sample(id, geom)
    VALUES ( :id, :shape_buffer );

```

---

## Grupo de transformación ST\_GML

Utilice el grupo de transformación ST\_GML para intercambiar datos con DB2<sup>®</sup> utilizando GML (Geography Markup Language (GML)).

Cuando se vincula un valor del servidor de bases de datos con el cliente, se utiliza la misma función proporcionada por ST\_AsGML() para convertir una geometría en su representación GML. Cuando se transfiere la representación GML de una geometría al servidor de bases de datos, se utiliza implícitamente la función ST\_Geometry(CLOB) para realizar las conversiones a un valor de tipo ST\_Geometry. El uso del grupo de transformación para vincular valores con DB2 hace que las geometrías se representen en el sistema de referencia espacial con el identificador numérico 0 (cero).

### Ejemplos

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

#### Ejemplo 1

El siguiente script de SQL muestra el uso del grupo de transformación ST\_GML para obtener una geometría en su representación GML sin utilizar la función explícita ST\_AsGML.

```

CREATE TABLE transforms_sample (
    id INTEGER,
    geom db2gse.ST_Geometry)

INSERT
    INTO transforms_sample
    VALUES ( 1, db2gse.ST_Geometry('multipoint z (10 10
        3, 20 20 4, 15 20 30)', 0) )
    SET CURRENT DEFAULT TRANSFORM GROUP = ST_GML

SELECT id, geom
FROM transforms_sample
WHERE id = 1

```

Resultados:

```

ID      GEOM
-----
1 <gml:MultiPoint srsName=UNSPECIFIED><gml:PointMember>
  <gml:Point><gml:coord><gml:X>10</gml:X>

```

```

<gml:Y>10</gml:Y><gml:Z>3</gml:Z>
</gml:coord></gml:Point></gml:PointMember>
<gml:PointMember><gml:Point><gml:coord>
<gml:X>20</gml:X><gml:Y>20</gml:Y>
<gml:Z>4</gml:Z></gml:coord></gml:Point>
</gml:PointMember><gml:PointMember><gml:Point>
<gml:coord><gml:X>15</gml:X><gml:Y>20
</gml:Y><gml:Z>30</gml:Z></gml:coord>
</gml:Point></gml:PointMember></gml:MultiPoint>

```

## Ejemplo 2

El siguiente código en C muestra cómo utilizar el grupo de transformación ST\_GML para insertar geometrías sin utilizar la función explícita ST\_Geometry para la variable de lenguaje principal gml\_buffer, cuyo tipo es CLOB y que contiene la representación GML del punto (20 ,20) que se debe insertar.

```

EXEC SQL BEGIN DECLARE SECTION;
    sqlint32 id = 0;
    SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) gml_buffer;
EXEC SQL END DECLARE SECTION;

// definir el grupo de transformación para todas las sentencias de SQL
// posteriores
EXEC SQL
    SET CURRENT DEFAULT TRANSFORM GROUP = ST_GML;
    id = 100;
strcpy(gml_buffer.data, "<gml:point><gml:coord>"
    "<gml:X>20</gml:X> <gml:Y>20</gml:Y></gml:coord></gml:point>");

// inicializar variables de lenguaje principal
wkt_buffer.length = strlen(gml_buffer.data);

// insertar punto con WKT en la columna de tipo ST_Geometry
EXEC SQL
    INSERT
    INTO transforms_sample(id, geom)
    VALUES ( :id, :gml_buffer );

```

---

## Capítulo 25. Formatos de datos soportados

Este capítulo describe los formatos de datos espaciales industriales aceptados que se pueden utilizar con DB2 Spatial Extender. Se describen estos cuatro formatos de datos espaciales:

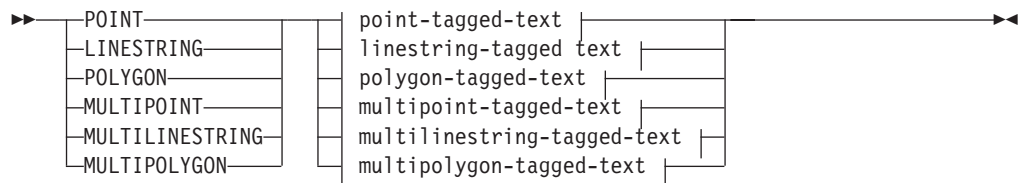
- Representación de texto convencional (well-known text, WKT)
- Representación binaria convencional (well-known binary, WKB)
- Representación de forma
- Representación GML (Geography Markup Language)

---

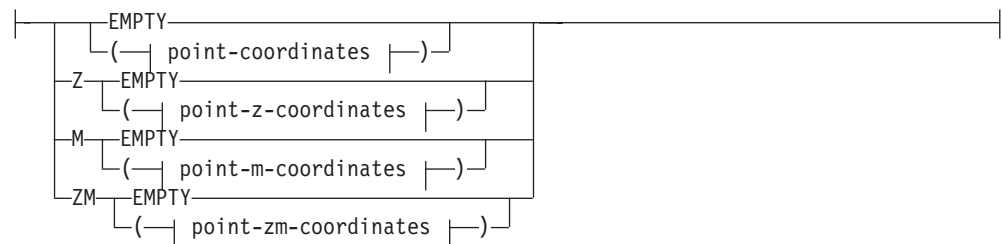
### Representación de texto convencional (well-known text, WKT)

La especificación "Simple Features for SQL" de OpenGIS Consortium define la representación de texto convencional para intercambiar datos de geometría en formato ASCII. Esta representación también está documentada en el estándar "SQL/MM Part: 3 Spatial" de ISO. Consulte "Funciones espaciales que convierten geometrías entre formatos de intercambio de datos" para obtener información sobre funciones que aceptan y producen datos en formato WKT.

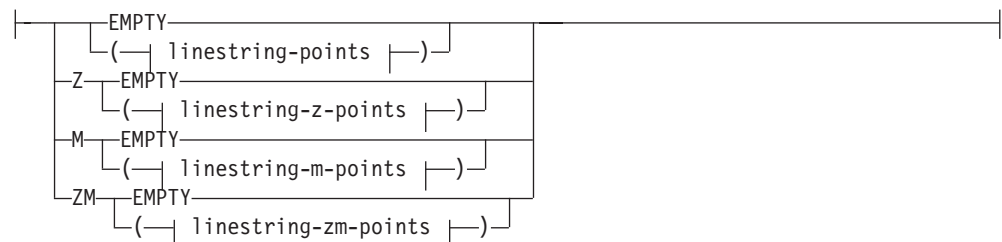
La representación de texto convencional de una geometría está definida de esta manera:



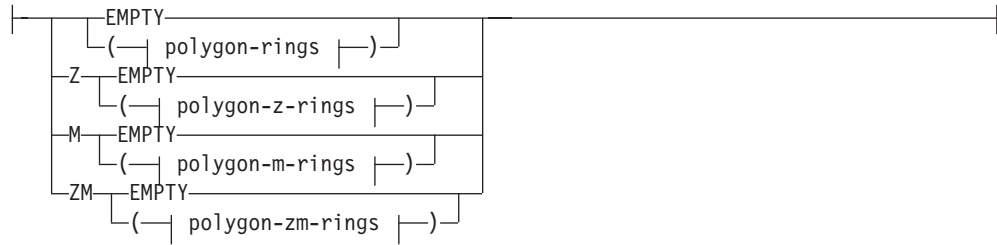
#### point-tagged-text:



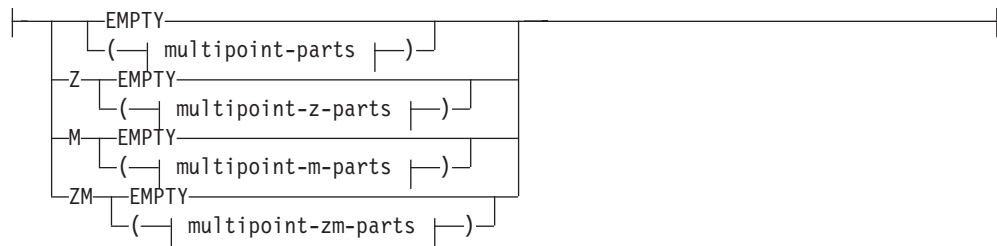
#### linestring-tagged-text:



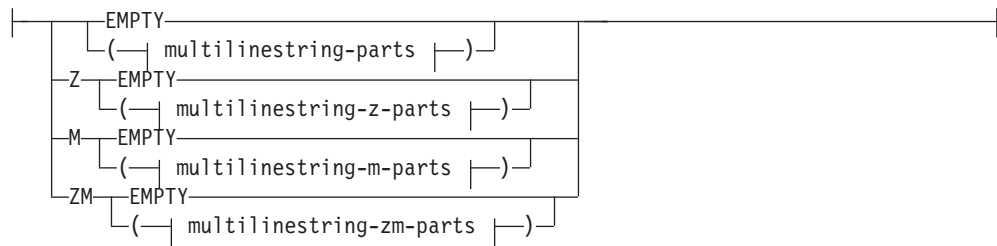
### **polygon-tagged-text:**



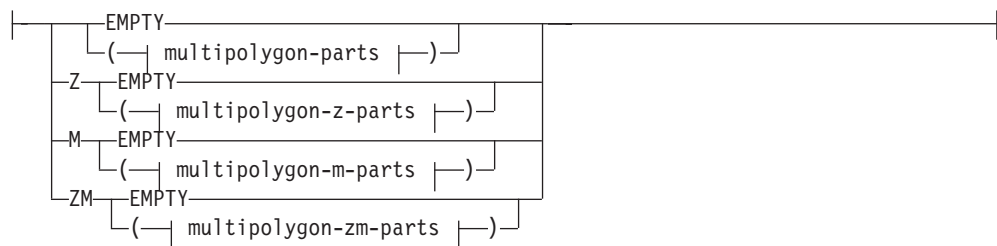
### **multipoint-tagged-text:**



### **multilinestring-tagged-text:**



### **multipolygon-tagged-text:**



### **point-coordinates:**



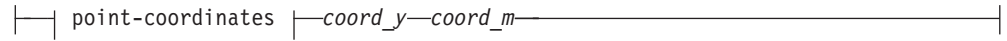
### **point-z-coordinates:**



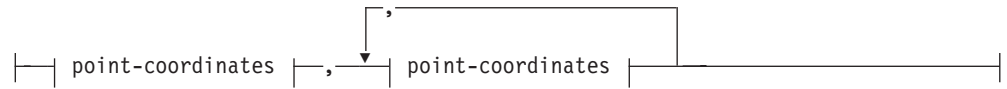
**point-m-coordinates:**



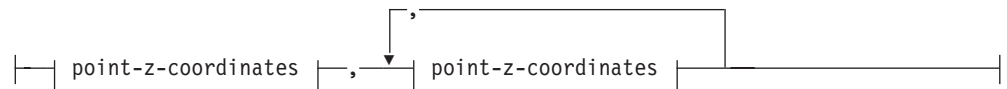
**point-zm-coordinates:**



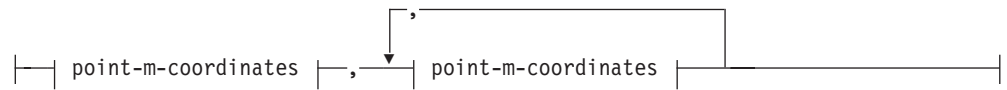
**linestring-points:**



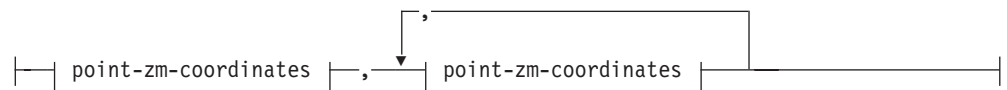
**linestring-z-points:**



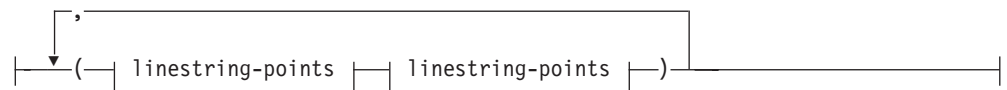
**linestring-m-points:**



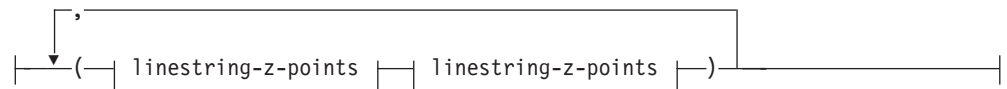
**linestring-zm-points:**



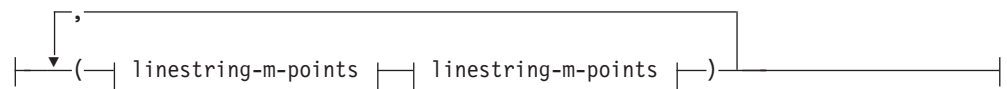
**polygon-rings:**



**polygon-z-rings:**



**polygon-m-rings:**



**polygon-zm-rings:**



**multipoint-parts:**



**multipoint-z-parts:**



**multipoint-m-parts:**



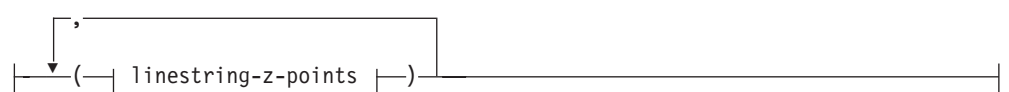
**multipoint-zm-parts:**



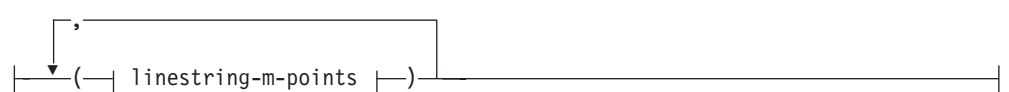
**multilinestring-parts:**



**multilinestring-z-parts:**



**multilinestring-m-parts:**





### **multilinestring-zm-parts:**



### **multipolygon-parts:**



### **multipolygon-z-parts:**



### **multipolygon-m-parts:**



### **multipolygon-zm-parts:**



## **Parámetros**

*coord\_x*

Valor numérico (fijo, entero o de coma flotante) que representa la coordenada X de un punto.

*coord\_y*

Valor numérico (fijo, entero o de coma flotante) que representa la coordenada Y de un punto.

*coord\_z*

Valor numérico (fijo, entero o de coma flotante) que representa la coordenada Z de un punto.

*coord\_m*

Valor numérico (fijo, entero o de coma flotante) que representa la coordenada M (medida) de un punto.

Si la geometría está vacía, se debe especificar la palabra clave EMPTY en lugar de la lista de coordenadas. La palabra clave EMPTY no debe estar incluida dentro de la lista de coordenadas.

La tabla siguiente proporciona algunos ejemplos de posibles representaciones de texto.

*Tabla 58. Tipos de geometría y sus representaciones de texto*

Tipo de geometría	Representación	Comentario
punto	POINT EMPTY	punto vacío
punto	POINT ( 10.05 10.28 )	punto
punto	POINT Z( 10.05 10.28 2.51 )	punto con coordenada Z
punto	POINT M( 10.05 10.28 4.72 )	punto con coordenada M
punto	POINT ZM( 10.05 10.28 2.51 4.72 )	punto con coordenada Z y coordenada M
cadena lineal	LINESTRING EMPTY	cadena lineal vacía
polígono	POLYGON (( 10 10, 10 20, 20 20, 20 15, 10 10))	polígono
multipunto	MULTIPOINT Z(10 10 2, 20 20 3)	multipunto con coordenadas Z
multilínea	MULTILINESTRING M(( 310 30 1, 40 30 20, 50 20 10 )( 10 10 0, 20 20 1))	multilínea con coordenadas M
multipolígono	MULTIPOLYGON ZM((( 1 1 1 1, 1 2 3 4, 2 2 5 6, 2 1 7 8, 1 1 1 1 )))	multipolígono con coordenadas Z y coordenadas M

## Representación binaria convencional (well-known binary, WKB)

Esta sección describe la representación binaria convencional de las geometrías.

La especificación "Simple Features for SQL" de OpenGIS Consortium define la representación binaria convencional. Esta representación también está definida en el estándar "SQL/MM Part: 3 Spatial" de ISO (International Organization for Standardization). Remítase a la sección de consulta correspondiente, al final de este tema, para obtener información sobre funciones que aceptan y producen datos en formato WKB.

El componente básico de la representación binaria convencional es la corriente de bytes correspondiente a un punto, que consta de dos valores de tipo "double". Las corrientes de bytes correspondientes a otras geometrías se crean utilizando las corrientes de bytes de geometrías que ya están definidas.

El ejemplo siguiente muestra el componente básico de las representaciones binarias convencionales.

```

// Basic Type definitions
// byte : 1 byte
// uint32 : 32 bit unsigned integer (4 bytes)
// double : double precision number (8 bytes)

// Building Blocks : Point, LinearRing

Point {
    double x;
    double y;
};
LinearRing {
    uint32 numPoints;
    Point points[numPoints];
};
enum wkbGeometryType {
    wkbPoint = 1,
    wkbLineString = 2,
    wkbPolygon = 3,
    wkbMultiPoint = 4,
    wkbMultiLineString = 5,
    wkbMultiPolygon = 6
};
enum wkbByteOrder {
    wkbXDR = 0, // Big Endian
    wkbNDR = 1 // Little Endian
};
WKBPoint {
    byte byteOrder;
    uint32 wkbType; // 1=wkbPoint
    Point point;
};
WKBLineString {
    byte byteOrder;
    uint32 wkbType; // 2=wkbLineString
    uint32 numPoints;
    Point points[numPoints];
};
WKBPolygon {
    byte byteOrder;
    uint32 wkbType; // 3=wkbPolygon
    uint32 numRings;
    LinearRing rings[numRings];
};
WKBMultiPoint {
    byte byteOrder;
    uint32 wkbType; // 4=wkbMultipoint
    uint32 num_wkbPoints;
    WKBPoint WKBPoints[num_wkbPoints];
};
WKBMultiLineString {
    byte byteOrder;
    uint32 wkbType; // 5=wkbMultiLineString
    uint32 num_wkbLineStrings;
    WKBLineString WKBLineStrings[num_wkbLineStrings];
};
wkbMultiPolygon {
    byte byteOrder;
    uint32 wkbType; // 6=wkbMultiPolygon
    uint32 num_wkbPolygons;
    WKBPolygon wkbPolygons[num_wkbPolygons];
};
WKBGeometry {
    union {

```

```

WKBPoint point;
WKBLineString linestring;
WKBPolygon polygon;
WKBMultiPoint mpoint;
WKBMultiLineString mlinestring;
WKBMultiPolygon mpolygon;
}
};

```

La figura siguiente es un ejemplo de geometría en representación binaria convencional que hace uso de la codificación NDR.

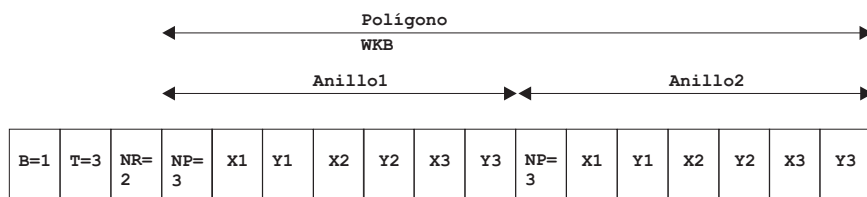


Figura 57. Representación de una geometría en formato NDR. (B=1) de tipo polígono (T=3) con 2 lineales (NR=2), donde cada anillo tiene 3 puntos (NP=3).

---

## Representación de forma

La representación de forma es un estándar de uso habitual definido por ESRI. Para obtener una descripción completa de la representación de forma, consulte el sitio web de ESRI situado en <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>.

---

## Representación GML (Geography Markup Language)

DB2 Spatial Extender tiene varias funciones que crean geometrías a partir de representaciones en formato GML (Geography Markup Language).

GML (Geography Markup Language) es una codificación XML para información geográfica que está definida por la especificación "Geography Markup Language V2" de OpenGIS Consortium. Esta especificación de OpenGIS Consortium se puede encontrar en este sitio Web: <http://www.opengis.org/techno/implementation.htm>.

---

## Capítulo 26. Sistemas de coordenadas soportados

Este tema describe la sintaxis de los sistemas de coordenadas y lista los valores de sistemas de coordenadas que están soportados por DB2 Spatial Extender.

---

### Sintaxis de los sistemas de coordenadas

La representación de texto convencional de los sistemas de referencia espacial proporciona una representación textual estándar para la información sobre sistemas de coordenadas. Las definiciones de la representación de texto convencional están definidas por la especificación de OGC "Simple Features for SQL" y por el estándar SQL/MM Part 3: Spatial de ISO.

Un sistema de coordenadas es un sistema de coordenadas geográficas (latitud-longitud), un sistema de coordenadas proyectadas (X,Y) o un sistema de coordenadas geocéntricas (X,Y,Z). El sistema de coordenadas consta de varios objetos. Cada objeto tiene asignada una palabra clave escrita en mayúsculas (por ejemplo, DATUM o UNIT) seguida entre paréntesis o corchetes por los parámetros que definen el objeto, separados por comas. Algunos objetos están formados por otros objetos, por lo que el resultado es una estructura anidada.

**Nota:** Las implementaciones pueden utilizar paréntesis estándar ( ) en vez de corchetes [ ] y debe ser posible leer ambos tipos de símbolos.

La definición EBNF (Extended Backus Naur Form) para la representación en forma de cadena de caracteres de un sistema de coordenadas utilizando corchetes es la siguiente (consulte la nota anterior sobre el uso de corchetes):

```
<coordinate system> = <projected cs> |  
<geographic cs> | <geocentric cs>  
<projected cs> = PROJCS["<name>",  
<geographic cs>, <projection>, {<parameter>,*  
<linear unit>]  
<projection> = PROJECTION["<name>"]  
<parameter> = PARAMETER["<name>",  
<value>]  
  
<value> = <number>
```

El tipo de sistema de coordenadas está identificado por la palabra clave utilizada:

#### **PROJCS**

Esta palabra clave identifica un sistema de coordenadas de un conjunto de datos cuando los datos están en coordenadas proyectadas

#### **GEOGCS**

Esta palabra clave identifica un sistema de coordenadas de un conjunto de datos cuando los datos están en coordenadas geográficas

#### **GEOCCS**

Esta palabra clave identifica un sistema de coordenadas de un conjunto de datos cuando los datos están en coordenadas geocéntricas

La palabra clave PROJCS va seguida por todos los "elementos" que definen el sistema de coordenadas proyectadas. El primer elemento de cualquier objeto es siempre el nombre. El nombre del sistema de coordenadas proyectadas va seguido por varios objetos: el sistema de coordenadas geográficas, la proyección

cartográfica, uno o varios parámetros y la unidad lineal de medida. Todos los sistemas de coordenadas proyectadas están basados en un sistema de coordenadas geográficas, por lo que esta sección describe primero los elementos específicos de un sistema de coordenadas proyectadas. Por ejemplo, la zona 10N de UTM sobre el sistema de referencia NAD83 está definida así:

```
PROJCS["NAD_1983_UTM_Zone_10N",
<geographic cs>,
PROJECTION["Transverse_Mercator"],
PARAMETER["False_Easting",500000.0],
PARAMETER["False_Northing",0.0],
PARAMETER["Central_Meridian",-123.0],
PARAMETER["Scale_Factor",0.9996],
PARAMETER["Latitude_of_Origin",0.0],
UNIT["Meter",1.0]]
```

El nombre y varios objetos definen el objeto del sistema de coordenadas geográficas secuencialmente: el sistema de referencia, el meridiano de origen y la unidad angular de medida.

```
<geographic cs> = GEOGCS["<name>", <datum>, <prime meridian>, <angular unit>]
<datum> = DATUM["<name>", <spheroid>]
<spheroid> = SPHEROID["<name>", <semi-major axis>, <inverse flattening>]
<semi-major axis> = <number>
<inverse flattening> = <number>
<prime meridian> = PRIMEM["<name>", <longitude>]
<longitude> = <number>
```

El eje semimayor se mide en metros y debe ser mayor que cero.

Esta sería la definición del sistema de coordenadas geográficas para la zona 10 de UTM en NAD83:

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],
UNIT["Degree",0.0174532925199433]]
```

El objeto UNIT puede representar una unidad angular o lineal de medidas:

```
<angular unit> = <unit>
<linear unit> = <unit>
<unit> = UNIT["<name>", <conversion factor>]
<conversion factor> = <number>
```

El factor de conversión especifica el número de metros (para una unidad lineal) o el número de radianes (para una unidad angular) por cada unidad y debe ser mayor que cero.

Por tanto, la representación completa en forma de cadena de caracteres de la zona 10N de UTM sería la siguiente:

```
PROJCS["NAD_1983_UTM_Zone_10N",
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433]],
PROJECTION["Transverse_Mercator"],PARAMETER["False_Easting",500000.0],
PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",-123.0],
PARAMETER["Scale_Factor",0.9996],PARAMETER["Latitude_of_Origin",0.0],
UNIT["Meter",1.0]]
```

Un sistema de coordenadas geocéntricas es bastante similar a un sistema de coordenadas geográficas:

```
<geocentric cs> = GEOCCS["<name>", <datum>, <prime meridian>, <linear unit>]
```

## Unidades lineales soportadas

Tabla 59. Unidades lineales soportadas

Unidad	Factor de conversión
Metro	1,0
Pie (Internacional)	0,3048
Pie americano	12/39,37
Pie americano modificado	12,0004584/39,37
Pie de Clarke	12/39,370432
Pie indio	12/39,370141
Link	7,92/39,370432
Link (Benoit)	7,92/39,370113
Link (Sears)	7,92/39,370147
Chain (Benoit)	792/39,370113
Chain (Sears)	792/39,370147
Yarda (India)	36/39,370141
Yarda (Sears)	36/39,370147
Braza	1,8288
Milla náutica	1852,0

## Unidades angulares soportadas

Tabla 60. Unidades angulares soportadas

Unidad	Rango válido para la latitud	Rango válido para la longitud	Factor de conversión
Radián	Radianes $-\pi/2$ y $\pi/2$ (inclusive)	Radianes $-\pi$ y $\pi$ (inclusive)	1,0
Grado decimal	$-90$ y $90$ grados (inclusive)	$-180$ y $180$ grados (inclusive)	$\pi/180$
Minuto decimal	$-5400$ y $5400$ minutos (inclusive)	$-10800$ y $10800$ minutos (inclusive)	$(\pi/180)/60$
Segundo decimal	$-324000$ y $324000$ segundos (inclusive)	$-648000$ y $648000$ segundos (inclusive)	$(\pi/180)*3600$
Gon	$-100$ y $100$ gradianes (inclusive)	$-200$ y $200$ gradianes (inclusive)	$\pi/200$
Grad	$-100$ y $100$ gradianes (inclusive)	$-200$ y $200$ gradianes (inclusive)	$\pi/200$

## Esferoides soportados

Tabla 61. Esferoides soportados

Nombre	Eje semimayor	Aplanamiento inverso
Airy 1830	6377563,396	299,3249646
Airy Modified 1849	6377340,189	299,3249646
Average Terrestrial System 1977	6378135,0	298,257
Australian National Spheroid	6378160,0	298,25
Bessel 1841	6377397,155	299,1528128
Bessel Modified	6377492,018	299,1528128
Bessel Namibia	6377483,865	299,1528128
Clarke 1858	6378293.639	294.260676369
Clarke 1866	6378206,4	294,9786982
Clarke 1866 (Michigan)	6378450,047	294,978684677
Clarke 1880	6378249,138	293,466307656
Clarke 1880 (Arc)	6378249,145	293,466307656
Clarke 1880 (Benoit)	6378300,79	293,466234571
Clarke 1880 (IGN)	6378249,2	293,46602
Clarke 1880 (RGS)	6378249,145	293,465
Clarke 1880 (SGA 1922)	6378249,2	293,46598
Everest (1830 Definition)	6377299,36	300,8017
Everest 1830 Modified	6377304,063	300,8017
Everest Adjustment 1937	6377276,345	300,8017
Everest 1830 (1962 Definition)	6377301,243	300,8017255
Everest 1830 (1967 Definition)	6377298,556	300,8017
Everest 1830 (1975 Definition)	6377299,151	300,8017255
Everest 1969 Modified	6377295,664	300,8017
Fischer 1960	6378166,0	298,3
Fischer 1968	6378150,0	298,3
Modified Fischer	6378155,0	298,3
GEM 10C	6378137,0	298,257222101
GRS 1967	6378160,0	298,247167427



Tabla 61. Esferoides soportados (continuación)

Nombre	Eje semimayor	Aplanamiento inverso
GRS 1967 Truncated	6378160,0	298,25
GRS 1980	6378137,0	298,257222101
Helmert 1906	6378200,0	298,3
Hough 1960	6378270,0	297,0
Indonesian National Spheroid	6378160,0	298,247
International 1924	6378388,0	297,0
International 1967	6378160,0	298,25
Krassowsky 1940	6378245,0	298,3
NWL 9D	6378145,0	298,25
NWL 10D	6378135,0	298,26
OSU 86F	6378136,2	298,25722
OSU 91A	6378136,3	298,25722
Plessis 1817	6376523,0	308,64
Sphere	6371000,0	0,0
Sphere (ArcInfo)	6370997,0	0,0
Struve 1860	6378298,3	294,73
Walbeck	6376896,0	302,78
War Office	6378300,0	296,0
WGS 1966	6378145,0	298,25
WGS 1972	6378135,0	298,26
WGS 1984	6378137,0	298,257223563

## Sistemas de referencia geodésicos soportados

Tabla 62. Sistemas de referencia geodésicos soportados

Nombre	Sistema de referencia geodésico
Adindan	Lisbon (Lisboa)
Afgooye	Loma Quintana
Agadez	Lome
Australian Geodetic Datum 1966	Luzon 1911
Australian Geodetic Datum 1984	Mahe 1971

*Tabla 62. Sistemas de referencia geodésicos soportados (continuación)*

<b>Nombre</b>	<b>Sistema de referencia geodésico</b>
Ain el Abd 1970	Makassar
Amersfoort	Malongo 1987
Aratu	Manoca
Arc 1950	Massawa
Arc 1960	Merchich
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko

Tabla 62. Sistemas de referencia geodésicos soportados (continuación)

Nombre	Sistema de referencia geodésico
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Bogota	Nord de Guerre
Bukit Rimpah	NGO 1948
Camacupa	Nord Sahara 1959
Campo Inchauspe	NSWC 9Z-2
Cape	Nouvelle Triangulation Francaise
Carthage	New Zealand Geodetic Datum 1949
Chua	OS (SN) 1980
Conakry 1905	OSGB 1936
Corrego Alegre	OSGB 1970 (SN)
Cote d'Ivoire	Padang 1884
Datum 73	Palestine 1923
Deir ez Zor	Pointe Noire
Deutsche Hauptdreiecksnetz	Provisional South American Datum 1956
Douala	Pulkovo 1942
European Datum 1950	Qatar
European Datum 1987	Qatar 1948
Egypt 1907	Qornoq
European Reference System 1989	RT38
Fahud	South American Datum 1969
Gandajika 1970	Sapper Hill 1943
Garoua	Schwarzeck
Geocentric Datum of Australia 1994	Segora
Guyane Francaise	Serindung
Herat North	Stockholm 1938
Hito XVIII 1963	Sudan

Tabla 62. Sistemas de referencia geodésicos soportados (continuación)

Nombre	Sistema de referencia geodésico
Hu Tzu	Shan Tananarive 1925
Hungarian Datum 1972	Timbalai 1948
Indian 1954	TM65
Indian 1975	TM75
Indonesian Datum 1974	Tokyo
Jamaica 1875	Trinidad 1903
Jamaica 1969	Trucial Coast 1948
Kalianpur	Voirol 1875
Kandawala	Voirol Unifie 1960
Kertau	WGS 1972
Kuwait Oil Company	WGS 1972 Transit Broadcast Ephemeris
La Canoa	WGS 1984
Lake	Yacare
Leigon	Yoff
Liberia 1964	Zanderij

## Meridianos de origen soportados

Tabla 63. Meridianos de origen soportados

Ubicación	Coordenadas
Greenwich	0° 0' 0"
Berna	7° 26' 22.5" E
Bogotá	74° 4' 51.3" W
Bruselas	4° 22' 4.71" E
Ferro	17° 40' 0" W
Yakarta	106° 48' 27.79" E
Lisboa	9° 7' 54.862" W
Madrid	3° 41' 16.58" W
París	2° 20' 14.025" E
Roma	12° 27' 8.4" E
Estocolmo	18° 3' 29" E

## Proyecciones cartográficas soportadas

Tabla 64. *Proyecciones cilíndricas*

Proyecciones cilíndricas	Proyecciones seudocilíndricas
Behrmann	Parabólica de Craster
Cassini	Eckert I
Cilíndrica de igual área	Eckert II
Equirrectangular	Eckert III
Estereográfica de Gall	Eckert IV
Gauss-Kruger	Eckert V
Mercator	Eckert VI
Cilíndrica de Miller	Cuártica polar plana de McBryde-Thomas
Oblicua	Mercator (Hotine) Mollweide
Plate-Carée	Robinson
Times	Sinusoidal (Sansom-Flamsteed)
Transversal de Mercator	Winkel I

Tabla 65. *Proyecciones cónicas*

Nombre	Proyección cónica
Cónica de igual área de Albers	Trimétrica de Chamberlin
Cónica conforme oblicua bipolar	Equidistante de dos puntos
Bonne	Hammer-Aitoff de igual área
Cónica equidistante	Van der Grinten I
Cónica conforme de Lambert	Varios
Policónica	Alaska serie E
Cónica simple	Alaska Grid (Estereográfica modificada por Snyder)

Tabla 66. *Parámetros de la proyección cartográfica*

Parámetro	Descripción
central_meridian	Línea de longitud elegida como origen de las coordenadas X.
scale_factor	Es un factor de escala que se utiliza generalmente para disminuir el grado de distorsión en una proyección cartográfica.

Tabla 66. Parámetros de la proyección cartográfica (continuación)

Parámetro	Descripción
standard_parallel_1	Línea de latitud que básicamente carece de distorsión. También se utiliza para "latitud de escala verdadera".
standard_parallel_2	Línea de longitud que básicamente carece de distorsión.
longitude_of_center	Longitud que define el punto central de la proyección cartográfica.
latitude_of_center	Latitud que define el punto central de la proyección cartográfica.
longitude_of_origin	Longitud elegida como origen de las coordenadas X.
latitude_of_origin	Latitud elegida como origen de las coordenadas Y.
false_easting	Valor que se agrega a las coordenadas X para que todos sus valores sean positivos.
false_northing	Valor que se agrega a las coordenadas Y para que todos sus valores sean positivos.
azimuth	Ángulo hacia el este del norte que define la línea central de una proyección oblicua.
longitude_of_point_1	Longitud del primer punto necesario para una proyección cartográfica.
latitude_of_point_1	Latitud del primer punto necesario para una proyección cartográfica.
longitude_of_point_2	Longitud del segundo punto necesario para una proyección cartográfica.
latitude_of_point_2	Latitud del segundo punto necesario para una proyección cartográfica.
longitude_of_point_3	Longitud del tercer punto necesario para una proyección cartográfica.
latitude_of_point_3	Latitud del tercer punto necesario para una proyección cartográfica.
landsat_number	Número de un satélite Landsat.
path_number	Número de la ruta orbital de un determinado satélite.
perspective_point_height	Altura sobre la superficie terrestre del punto de perspectiva de la proyección cartográfica.
fipszone	Número de zona del sistema de coordenadas de Plano de estado.
zone	Número de zona de UTM.

---

## Capítulo 27. Tareas espaciales desde el Centro de control de DB2

Las tareas que están disponibles en la interfaz de usuario de Spatial Extender pueden ayudar a simplificar tareas.

Puede ejecutar muchas tareas desde un indicador de mandatos en forma de mandato, desde una aplicación en forma de procedimiento almacenado o desde el Centro de control de DB2. Esta sección describe las tareas que se pueden realizar desde el Centro de control de DB2.

---

### Modificación de un sistema de coordenadas

La modificación de un sistema de coordenadas puede cambiar en gran medida la ubicación de la geometría y puede convertirla en inservible. Asegúrese de que comprende las ramificaciones de los cambios antes de modificar un sistema de coordenadas.

Puede cambiar cualquiera de los cambios, excepto **Nombre**.

- **Organización e ID de organización:** Estos valores son opcionales. Ambos deben ser nulos o bien no nulos. La combinación de los parámetros identifica de forma exclusiva el sistema de coordenadas.
- **Definición:** La definición puede contener hasta 2048 caracteres. El proveedor que suministra el sistema de coordenadas incluye generalmente este valor.

---

### Creación de un sistema de coordenadas

Normalmente, debe utilizar un sistema de coordenadas existente.

Si crea su propio sistema de coordenadas, compruebe que toda la información de coordenadas que está proporcionando sea preciso y compatible con el geonavegador que está utilizando.

- **Nombre:** Le ayuda a identificar el sistema de coordenadas. Organización e ID de organización identifican el sistema de coordenadas.
- **Organización e ID de organización:** Estos valores son opcionales. Ambos deben ser nulos o bien no nulos. La combinación de los parámetros identifica de forma exclusiva el sistema de coordenadas.
- **Definición:** La definición puede contener hasta 2048 caracteres. El proveedor que suministra el sistema de coordenadas incluye generalmente este valor.

---

### Creación de una columna espacial

Si añade datos espaciales a una tabla existente, debe crear una columna espacial y registrarla con un sistema de referencia espacial.

Para crear una columna espacial utilizando la ventana Crear columna espacial, debe abrir la ventana Columnas espaciales desde una tabla.

- **Columna:** Escriba un nombre para la columna.
- **Esquema de tipo y Nombre de tipo:** Seleccione valores de la lista.

- **Sistema de referencia espacial:** Opcional: Seleccione un sistema de referencia espacial de la lista. Para registrar la columna con un sistema de referencia espacial, especifique un sistema de referencia geodésico con un ID comprendido entre 2000000000 y 2000001000.

---

## Creación de un índice espacial

Puede crear un índice reticular espacial o un índice Voronoi geodésico en una columna espacial. Cuando cree un índice, puede mejorar el rendimiento haciendo que los datos sean más fáciles de localizar y recuperar para DB2.

**Recomendación:** Utilice el mismo sistema de coordenadas para todos los datos de una columna espacial en la que desee crear un índice para asegurarse de que el índice devuelve los resultados correctos. Puede registrar una columna espacial para restringir todos los datos de modo que utilicen el mismo sistema de referencia espacial y, por consiguiente, el mismo sistema de coordenadas.

- **Columna espacial:** Seleccione la columna en la que se creará el índice.
- **Retícula más fina:** Para crear un índice Voronoi, escriba -1. De lo contrario, escriba un número mayor que 0.
- **Retícula media:** Para crear un índice Voronoi, escriba -1. Puede escribir 0 si no desea utilizar una retícula media o bien escriba un número mayor que la retícula más fina.
- **Retícula gruesa:** Para crear un índice Voronoi, escriba el identificador de la estructura de celdas que desea utilizar (de 1 a 14). Debe utilizar una retícula media para utilizar una retícula más gruesa. Este valor debe ser mayor que la retícula media. Puede escribir 0 si no desea utilizar una retícula gruesa.

---

## Ejecución de la geocodificación

Utilice el cuaderno Ejecutar geocodificación en el Centro de control de DB2 para geocodificar datos espaciales en modalidad de proceso por lotes.

### Página Básico

Especifique el geocodificador y a continuación, seleccione una columna de resultados. El campo **Tipo de resultado** se completa automáticamente basándose en el geocodificador que especifique. Puede personalizar todavía más el trabajo de proceso por lotes especificando un ámbito de confirmación y una cláusula WHERE.

### Página Parámetros del geocodificador

Puede especificar un nombre o un valor de columna propio para un parámetro.

---

## Definición de la geocodificación

A la hora de definir la geocodificación, debe asociar una columna con un geocodificador que se utilizará para geocodificar los datos y debe configurar los parámetros de geocodificación correspondientes y otra información relevante que el geocodificador utilizará más adelante.

### Página Básico

Seleccione qué geocodificar que desea configurar. El campo **Tipo de resultado** se completa automáticamente basándose en el geocodificador que especifique. Puede



utilizar el campo **Ámbito de confirmación** para seleccionar el número de filas que se geocodificarán antes de realizar una confirmación.

**Consejo:** Puede seleccionar un geocodificador de la lista y seleccionar **Geocodificar automáticamente** para geocodificar automáticamente la columna espacial cuando se actualiza.

#### **Página Parámetros del geocodificador**

Puede especificar un nombre o un valor de columna propio para un parámetro.

---

## **Modificación de un sistema de referencia espacial**

**Precaución:** Si modifica algún atributo distinto de la descripción, los valores de todos los datos espaciales asociados a dicho sistema de referencia espacial cambiarán y puede que ya no sean válidos.

- En el campo **ID**, escriba el ID del sistema de referencia espacial que está modificando. Para un sistema de referencia espacial geodésico, sólo puede cambiar los valores ID comprendidos entre 2000000318 y 2000001000.
- **Desplazamiento:** En el caso de las transformaciones de desplazamiento, escriba los valores de escala y desplazamiento para las coordenadas X, Y, Z o M. Un valor de escala es el valor multiplicador de las medidas. El valor por omisión es 1. Puede cambiar el valor para mantener la precisión cuando una coordenada se convierte desde un valor decimal a un valor entero. Un valor de desplazamiento es el número que se restará de cada valor para obtener un entero positivo para los valores de coordenadas y medidas. Los valores de escala y desplazamiento son necesarios para crear un sistema de referencia espacial.
- **Extensión:** En el caso de transformaciones, escriba los valores mínimo y máximo para las coordenadas X, Y, Z o M. Un valor de escala es el valor multiplicador de las medidas. El valor por omisión es 1. Puede cambiar el valor para mantener la precisión cuando una coordenada se convierte desde un valor decimal a un valor entero. Los valores de escala y extensión sólo son necesarios cuando se selecciona el botón de selección **Extensión**.

---

## **Importación de datos espaciales**

Puede importar archivos de formas.

#### **Página Básico**

Utilice esta página para especificar la información de origen y de destino. También puede especificar archivos de excepción y mensaje, que pueden ser útiles para resolver problemas en caso de que la importación no se realice correctamente.

#### **Página Avanzado**

Utilice esta página para especificar detalles del espacio de tabla y detalles sobre cómo debe realizarse la importación.



---

## Apéndice A. Visión general de la información técnica de DB2

La información técnica de DB2 está disponible a través de las herramientas y los métodos siguientes:

- Centro de información de DB2
  - Temas (Tareas, concepto y temas de consulta)
  - Ayuda para herramientas de DB2
  - Programas de ejemplo
  - Guías de aprendizaje
- Manuales de DB2
  - Archivos PDF (descargables)
  - Archivos PDF (desde el DVD con PDF de DB2)
  - Manuales en copia impresa
- Ayuda de línea de mandatos
  - Ayuda de mandatos
  - Ayuda de mensajes

**Nota:** Los temas del Centro de información de DB2 se actualizan con más frecuencia que los manuales en PDF o impresos. Para obtener la información más actualizada, instale las actualizaciones de la documentación cuando estén disponibles, o consulte el Centro de información de DB2 en [ibm.com](http://ibm.com).

Puede acceder a información técnica adicional de DB2 como, por ejemplo, notas técnicas, documentos técnicos y publicaciones IBM Redbooks en línea, en el sitio [ibm.com](http://ibm.com). Acceda al sitio de la biblioteca de software de gestión de información de DB2 en <http://www.ibm.com/software/data/sw-library/>.

### Comentarios sobre la documentación

Agradecemos los comentarios sobre la documentación de DB2. Si tiene sugerencias sobre cómo podemos mejorar la documentación de DB2, envíe un correo electrónico a [db2docs@ca.ibm.com](mailto:db2docs@ca.ibm.com). El personal encargado de la documentación de DB2 lee todos los comentarios de los usuarios, pero no puede responderlos directamente. Proporcione ejemplos específicos siempre que sea posible de manera que podamos comprender mejor sus problemas. Si realiza comentarios sobre un tema o archivo de ayuda determinado, incluya el título del tema y el URL.

No utilice esta dirección de correo electrónico para contactar con el Soporte al cliente de DB2. Si tiene un problema técnico de DB2 que no está tratado por la documentación, consulte al centro local de servicio técnico de IBM para obtener ayuda.

---

## Biblioteca técnica de DB2 en copia impresa o en formato PDF

Las tablas siguientes describen la biblioteca de DB2 que está disponible en el Centro de publicaciones de IBM en [www.ibm.com/shop/publications/order](http://www.ibm.com/shop/publications/order). Los manuales de DB2 Versión 9.7 en inglés y las versiones traducidas en formato PDF se pueden descargar del sitio web [www.ibm.com/support/docview.wss?rs=71&uid=swg2700947](http://www.ibm.com/support/docview.wss?rs=71&uid=swg2700947).

Aunque las tablas identifican los manuales en copia impresa disponibles, puede que dichos manuales no estén disponibles en su país o región.

El número de documento se incrementa cada vez que se actualiza un manual. Asegúrese de que lee la versión más reciente de los manuales, tal como aparece a continuación:

**Nota:** El Centro de información de DB2 se actualiza con más frecuencia que los manuales en PDF o impresos.

*Tabla 67. Información técnica de DB2*

<b>Nombre</b>	<b>Número de documento</b>	<b>Copia impresa disponible</b>	<b>Última actualización</b>
<i>Consulta de las API administrativas</i>	SC11-3912-00	Sí	Agosto de 2009
<i>Rutinas y vistas administrativas</i>	SC11-3909-00	No	Agosto de 2009
<i>Call Level Interface Guide and Reference, Volume 1</i>	SC27-2437-00	Sí	Agosto de 2009
<i>Call Level Interface Guide and Reference, Volume 2</i>	SC27-2438-00	Sí	Agosto de 2009
<i>Consulta de mandatos</i>	SC11-3914-00	Sí	Agosto de 2009
<i>Data Movement Utilities Guide and Reference</i>	SC27-2440-00	Sí	Agosto de 2009
<i>Data Recovery and High Availability Guide and Reference</i>	SC27-2441-00	Sí	Agosto de 2009
<i>Database Administration Concepts and Configuration Reference</i>	SC27-2442-00	Sí	Agosto de 2009
<i>Database Monitoring Guide and Reference</i>	SC27-2458-00	Sí	Agosto de 2009
<i>Database Security Guide</i>	SC27-2443-00	Sí	Agosto de 2009
<i>Guía de DB2 Text Search</i>	SC11-3927-00	Sí	Agosto de 2009
<i>Developing ADO.NET and OLE DB Applications</i>	SC27-2444-00	Sí	Agosto de 2009
<i>Developing Embedded SQL Applications</i>	SC27-2445-00	Sí	Agosto de 2009
<i>Desarrollo de aplicaciones Java</i>	SC11-3907-00	Sí	Agosto de 2009
<i>Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails</i>	SC11-3908-00	No	Agosto de 2009
<i>Developing User-defined Routines (SQL and External)</i>	SC27-2448-00	Sí	Agosto de 2009
<i>Getting Started with Database Application Development</i>	GI11-9410-00	Sí	Agosto de 2009

Tabla 67. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Última actualización
<i>Iniciación a la instalación y administración de DB2 en Linux y Windows</i>	GI11-8640-00	Sí	Agosto de 2009
<i>Globalization Guide</i>	SC27-2449-00	Sí	Agosto de 2009
<i>Instalación de servidores DB2</i>	SC11-3916-00	Sí	Agosto de 2009
<i>Instalación de clientes de servidor de datos de IBM</i>	SC11-3917-00	No	Agosto de 2009
<i>Consulta de mensajes Volumen 1</i>	SC11-3922-00	No	Agosto de 2009
<i>Consulta de mensajes Volumen 2</i>	SC11-3923-00	No	Agosto de 2009
<i>Net Search Extender Guía de administración y del usuario</i>	SC11-3926-00	No	Agosto de 2009
<i>Partitioning and Clustering Guide</i>	SC27-2453-00	Sí	Agosto de 2009
<i>pureXML Guide</i>	SC27-2465-00	Sí	Agosto de 2009
<i>Query Patroller Administration and User's Guide</i>	SC27-2467-00	No	Agosto de 2009
<i>Spatial Extender and Geodetic Data Management Feature Guía del usuario y manual de consulta</i>	SC11-3925-00	No	Agosto de 2009
<i>SQL Procedural Languages: Application Enablement and Support</i>	SC27-2470-00	Sí	Agosto de 2009
<i>Consulta de SQL, Volumen 1</i>	SC11-3910-00	Sí	Agosto de 2009
<i>Consulta de SQL, Volumen 2</i>	SC11-3911-00	Sí	Agosto de 2009
<i>Troubleshooting and Tuning Database Performance</i>	SC27-2461-00	Sí	Agosto de 2009
<i>Actualización a DB2 Versión 9.7</i>	SC11-3915-00	Sí	Agosto de 2009
<i>Guía de aprendizaje de Visual Explain</i>	SC11-3924-00	No	Agosto de 2009
<i>Novedades en DB2 Versión 9.7</i>	SC11-3921-00	Sí	Agosto de 2009
<i>Workload Manager Guide and Reference</i>	SC27-2464-00	Sí	Agosto de 2009
<i>XQuery Reference</i>	SC27-2466-00	No	Agosto de 2009

Tabla 68. Información técnica específica de DB2 Connect

Nombre	Número de documento	Copia impresa disponible	Última actualización
<i>Instalación y configuración de DB2 Connect Personal Edition</i>	SC11-3919-00	Sí	Agosto de 2009
<i>Instalación y configuración de servidores DB2 Connect</i>	SC11-3920-00	Sí	Agosto de 2009
<i>Guía del usuario de DB2 Connect</i>	SC11-3918-00	Sí	Agosto de 2009

Tabla 69. Información técnica de Information Integration

Nombre	Número de documento	Copia impresa disponible	Última actualización
<i>Information Integration: Administration Guide for Federated Systems</i>	SC19-1020-02	Sí	Agosto de 2009
<i>Information Integration: ASNCLP Program Reference for Replication and Event Publishing</i>	SC11-3900-04	Sí	Agosto de 2009
<i>Information Integration: Configuration Guide for Federated Data Sources</i>	SC19-1034-02	No	Agosto de 2009
<i>Information Integration: SQL Replication Guide and Reference</i>	SC11-3899-02	Sí	Agosto de 2009
<i>Information Integration: Introduction to Replication and Event Publishing</i>	GC19-1028-02	Sí	Agosto de 2009

## Pedido de manuales de DB2 en copia impresa

Si necesita manuales de DB2 en copia impresa, puede comprarlos en línea en varios países o regiones, pero no en todos. Siempre puede hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM. Recuerde que algunas publicaciones en copia software del DVD *Documentación en PDF de DB2* no están disponibles en copia impresa. Por ejemplo, no está disponible la publicación *Consulta de mensajes de DB2* en copia impresa.

Las versiones impresas de muchas de las publicaciones de DB2 disponibles en el DVD de Documentación en PDF de DB2 se pueden solicitar a IBM por una cantidad. Dependiendo desde dónde realice el pedido, podrá solicitar manuales en línea, desde el Centro de publicaciones de IBM. Si la realización de pedidos en línea no está disponible en su país o región, siempre puede hacer pedidos de manuales de DB2 en copia impresa al representante local de IBM. Tenga en cuenta que no todas las publicaciones del DVD de Documentación en PDF de DB2 están disponibles en copia impresa.

**Nota:** La documentación más actualizada y completa de DB2 se conserva en el Centro de información de DB2 en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7>.

Para hacer pedidos de manuales de DB2 en copia impresa:

- Para averiguar si puede hacer pedidos de manuales de DB2 en copia impresa en línea en su país o región, consulte el Centro de publicaciones de IBM en el sitio <http://www.ibm.com/shop/publications/order>. Debe seleccionar un país, región o idioma para poder acceder a la información sobre pedidos de publicaciones y, a continuación, seguir las instrucciones sobre pedidos para su localidad.
- Para hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM:
  1. Localice la información de contacto de su representante local desde uno de los siguientes sitios Web:
    - El directorio de IBM de contactos en todo el mundo en el sitio [www.ibm.com/planetwide](http://www.ibm.com/planetwide)
    - El sitio Web de publicaciones de IBM en el sitio <http://www.ibm.com/shop/publications/order>. Tendrá que seleccionar su país, región o idioma para acceder a la página de presentación de las publicaciones apropiadas para su localidad. Desde esta página, siga el enlace "Acerca de este sitio".
  2. Cuando llame, indique que desea hacer un pedido de una publicación de DB2.
  3. Proporcione al representante los títulos y números de documento de las publicaciones que desee solicitar. Si desea consultar los títulos y los números de documento, consulte el apartado "Biblioteca técnica de DB2 en copia impresa o en formato PDF" en la página 505.

---

## Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos

Los productos DB2 devuelven un valor de SQLSTATE para las condiciones que pueden ser el resultado de una sentencia de SQL. La ayuda de SQLSTATE explica los significados de los estados de SQL y los códigos de las clases de estados de SQL.

Para iniciar la ayuda para estados de SQL, abra el procesador de línea de mandatos y entre:

```
? sqlstate o ? código de clase
```

donde *sqlstate* representa un estado de SQL válido de cinco dígitos y *código de clase* representa los dos primeros dígitos del estado de SQL.

Por ejemplo, ? 08003 visualiza la ayuda para el estado de SQL 08003, y ? 08 visualiza la ayuda para el código de clase 08.

---

## Acceso a diferentes versiones del Centro de información de DB2

Para los temas de la versión 9.7 de DB2, el URL del Centro de información de DB2 es <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7>

Para los temas de DB2 Versión 9.5, el URL del Centro de información de DB2 es <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/>

Para los temas de DB2 Versión 9, el URL del Centro de información de DB2 es <http://publib.boulder.ibm.com/infocenter/db2luw/v9/>

Para los temas de DB2 Versión 8, vaya al URL del Centro de información de la Versión 8 en el sitio: <http://publib.boulder.ibm.com/infocenter/db2luw/v8/>

---

## Visualización de temas en su idioma preferido en el Centro de información de DB2

El Centro de información de DB2 intenta visualizar los temas en el idioma especificado en las preferencias del navegador. Si un tema no se ha traducido al idioma preferido, el Centro de información de DB2 visualiza dicho tema en inglés.

- Para visualizar temas en su idioma preferido en el navegador Internet Explorer:
  1. En Internet Explorer, pulse en el botón **Herramientas** —> **Opciones de Internet** —> **Idiomas...** Se abrirá la ventana Preferencias de idioma.
  2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
    - Para añadir un nuevo idioma a la lista, pulse el botón **Agregar...**

**Nota:** La adición de un idioma no garantiza que el sistema tenga los fonts necesarios para visualizar los temas en el idioma preferido.
    - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
  3. Borre la antememoria del navegador y, a continuación, renueve la página para visualizar el Centro de información de DB2 en su idioma preferido.
- Para visualizar temas en su idioma preferido en un navegador Firefox o Mozilla:
  1. Seleccione el botón en la sección **Idiomas** del diálogo **Herramientas** —> **Opciones** —> **Avanzado**. Se visualizará el panel Idiomas en la ventana Preferencias.
  2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
    - Para añadir un nuevo idioma a la lista, pulse el botón **Añadir...** a fin de seleccionar un idioma en la ventana Añadir idiomas.
    - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
  3. Borre la antememoria del navegador y, a continuación, renueve la página para visualizar el Centro de información de DB2 en su idioma preferido.

En algunas combinaciones de navegador y sistema operativo, también debe cambiar los valores regionales del sistema operativo al entorno local y al idioma de su elección.

---

## Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

El Centro de información de DB2 instalado en local se debe actualizar periódicamente.

### Antes de empezar



Ya debe haber un Centro de información de DB2 Versión 9.7 instalado. Para obtener información adicional, consulte el tema “Instalación del Centro de información de DB2 utilizando el Asistente de instalación de DB2” en la publicación *Instalación de servidores DB2*. Todos los requisitos previos y las restricciones aplicables a la instalación del Centro de información se aplican también a la actualización del Centro de información.

### Acerca de esta tarea

Un Centro de información de DB2 existente se puede actualizar automática o manualmente:

- Actualizaciones automáticas: actualiza las funciones y los idiomas del Centro de información existentes. Una ventaja adicional de las actualizaciones automáticas es que el Centro de información deja de estar disponible durante un período de tiempo mínimo mientras se realiza la actualización. Además, la ejecución de las actualizaciones automáticas se puede configurar como parte de otros trabajos de proceso por lotes que se ejecutan periódicamente.
- Actualizaciones manuales: se deben utilizar si se quieren añadir funciones o idiomas durante el proceso de actualización. Por ejemplo, un Centro de información en local se instaló inicialmente tanto en inglés como en francés, y ahora se desea instalar el idioma alemán. Con la actualización manual, se instalará el alemán y se actualizarán además las funciones y los idiomas existentes del Centro de información. No obstante, la actualización manual requiere que el usuario detenga, actualice y reinicie manualmente el Centro de información. El Centro de información no está disponible durante todo el proceso de actualización.

### Procedimiento

Este tema detalla el proceso de las actualizaciones automáticas. Para conocer las instrucciones para la actualización manual, consulte el tema “Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet”.

Para actualizar automáticamente el Centro de información de DB2 instalado en el sistema o en el servidor de Intranet:

1. En sistemas operativos Linux,
  - a. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio `/opt/ibm/db2ic/V9.7`.
  - b. Navegue desde el directorio de instalación al directorio `doc/bin`.
  - c. Ejecute el script `ic-update`:  
`ic-update`
2. En sistemas operativos Windows,
  - a. Abra una ventana de mandatos.
  - b. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio `<Archivos de programa>\IBM\Centro de información de DB2\Versión 9.7`, siendo `<Archivos de programa>` la ubicación del directorio Archivos de programa.
  - c. Navegue desde el directorio de instalación al directorio `doc\bin`.
  - d. Ejecute el archivo `ic-update.bat`:  
`ic-update.bat`

## Resultados

El Centro de información de DB2 se reinicia automáticamente. Si hay actualizaciones disponibles, el Centro de información muestra los temas nuevos y actualizados. Si no había actualizaciones del Centro de información disponibles, se añade un mensaje al archivo de anotaciones cronológicas. El archivo de anotaciones cronológicas está ubicado en el directorio `doc\eclipse\configuration`. El nombre del archivo de anotaciones cronológicas es un número generado aleatoriamente. Por ejemplo, `1239053440785.log`.

---

## Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

Si ha instalado localmente el Centro de información de DB2, puede obtener las actualizaciones de la documentación de IBM e instalarlas.

Para actualizar manualmente el Centro de información de DB2 instalado localmente es preciso que:

1. Detenga el Centro de información de DB2 en el sistema, y reinicie el Centro de información en modalidad autónoma. La ejecución del Centro de información en modalidad autónoma impide que otros usuarios de la red accedan al Centro de información y permite al usuario aplicar las actualizaciones. La versión para estaciones de trabajo del Centro de información de DB2 siempre se ejecuta en modalidad autónoma.
2. Utilice la función Actualizar para ver qué actualizaciones están disponibles. Si hay actualizaciones que debe instalar, puede utilizar la función Actualizar para obtenerlas y actualizarlas.

**Nota:** Si su entorno requiere la instalación de actualizaciones del Centro de información de DB2 en una máquina no conectada a Internet, duplique el sitio de actualizaciones en un sistema de archivos local utilizando una máquina que esté conectada a Internet y tenga instalado el Centro de información de DB2. Si muchos usuarios en la red van a instalar las actualizaciones de la documentación, puede reducir el tiempo necesario para realizar las actualizaciones duplicando también el sitio de actualizaciones localmente y creando un proxy para el sitio de actualizaciones.

Si hay paquetes de actualización disponibles, utilice la característica Actualizar para obtener los paquetes. Sin embargo, la característica Actualizar sólo está disponible en modalidad autónoma.

3. Detenga el Centro de información autónomo y reinicie el Centro de información de DB2 en su equipo.

**Nota:** En Windows 2008 y Windows Vista (y posterior), los mandatos listados más abajo deben ejecutarse como administrador. Para abrir un indicador de mandatos o una herramienta gráfica con privilegios de administrador completos, pulse con el botón derecho del ratón el atajo y, a continuación, seleccione **Ejecutar como administrador**.

Para actualizar el Centro de información de DB2 instalado en el sistema o en el servidor de Intranet:

1. Detenga el Centro de información de DB2.
  - En Windows, pulse **Inicio** → **Panel de control** → **Herramientas administrativas** → **Servicios**. A continuación, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Detener**.

- En Linux, especifique el mandato siguiente:  
`/etc/init.d/db2icdv97 stop`
2. Inicie el Centro de información en modalidad autónoma.
    - En Windows:
      - a. Abra una ventana de mandatos.
      - b. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio <Archivos de programa>\IBM\Centro de información de DB2\Versión 9.7, siendo <Archivos de programa> la ubicación del directorio Archivos de programa.
      - c. Navegue desde el directorio de instalación al directorio doc\bin.
      - d. Ejecute el archivo help\_start.bat:  
`help_start.bat`
    - En Linux:
      - a. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio /opt/ibm/db2ic/V9.7.
      - b. Navegue desde el directorio de instalación al directorio doc/bin.
      - c. Ejecute el script help\_start:  
`help_start`

Se abre el navegador Web por omisión de los sistemas para visualizar el Centro de información autónomo.

3. Pulse en el botón **Actualizar** (🔄). (JavaScript™ debe estar habilitado en el navegador.) En la derecha del panel del Centro de información, pulse en **Buscar actualizaciones**. Se visualiza una lista de actualizaciones para la documentación existente.
4. Para iniciar el proceso de instalación, compruebe las selecciones que desee instalar y, a continuación, pulse **Instalar actualizaciones**.
5. Cuando finalice el proceso de instalación, pulse **Finalizar**.
6. Detenga el Centro de información autónomo:
  - En Windows, navegue hasta el directorio doc\bin del directorio de instalación y ejecute el archivo help\_end.bat:  
`help_end.bat`

**Nota:** El archivo help\_end de proceso por lotes contiene los mandatos necesarios para detener sin peligro los procesos que se iniciaron mediante el archivo help\_start de proceso por lotes. No utilice Control-C ni ningún otro método para detener help\_start.bat.

  - En Linux, navegue hasta el directorio de instalación doc/bin y ejecute el script help\_end:  
`help_end`

**Nota:** El script help\_end contiene los mandatos necesarios para detener sin peligro los procesos que se iniciaron mediante el script help\_start. No utilice ningún otro método para detener el script help\_start.
7. Reinicie el Centro de información de DB2:
  - En Windows, pulse **Inicio** → **Panel de control** → **Herramientas administrativas** → **Servicios**. A continuación, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Iniciar**.
  - En Linux, especifique el mandato siguiente:

```
/etc/init.d/db2icdv97 start
```

El Centro de información de DB2 actualizado visualiza los temas nuevos y actualizados.

---

## Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 le ayudan a conocer diversos aspectos de productos DB2. Se proporcionan instrucciones paso a paso a través de lecciones.

### Antes de comenzar

Puede ver la versión XHTML de la guía de aprendizaje desde el Centro de información en el sitio <http://publib.boulder.ibm.com/infocenter/db2help/>.

Algunas lecciones utilizan datos o código de ejemplo. Consulte la guía de aprendizaje para obtener una descripción de los prerrequisitos para las tareas específicas.

### Guías de aprendizaje de DB2

Para ver la guía de aprendizaje, pulse el título.

#### **“pureXML” en *pureXML Guide***

Configure una base de datos DB2 para almacenar datos XML y realizar operaciones básicas con el almacén de datos XML nativos.

#### **“Visual Explain” en la *Guía de aprendizaje de Visual Explain***

Analizar, optimizar y ajustar sentencias de SQL para obtener un mejor rendimiento al utilizar Visual Explain.

---

## Información de resolución de problemas de DB2

Existe una gran variedad de información para la resolución y determinación de problemas para ayudarle en la utilización de productos de base de datos DB2.

### Documentación de DB2

Puede encontrar información sobre la resolución de problemas en la publicación *DB2 Troubleshooting Guide* o en la sección Conceptos fundamentales sobre bases de datos del Centro de información de DB2. En ellas encontrará información sobre cómo aislar e identificar problemas utilizando herramientas y programas de utilidad de diagnóstico de DB2, soluciones a algunos de los problemas más habituales y otros consejos sobre cómo solucionar problemas que podría encontrar en los productos DB2.

### Sitio web de soporte técnico de DB2

Consulte el sitio Web de soporte técnico de DB2 si tiene problemas y desea obtener ayuda para encontrar las causas y soluciones posibles. El sitio de soporte técnico tiene enlaces a las publicaciones más recientes de DB2, notas técnicas, Informes autorizados de análisis del programa (APAR o arreglos de defectos), fixpacks y otros recursos. Puede buscar en esta base de conocimiento para encontrar posibles soluciones a los problemas.

Acceda al sitio Web de soporte técnico de DB2 en la dirección [http://www.ibm.com/software/data/db2/support/db2\\_9/](http://www.ibm.com/software/data/db2/support/db2_9/)

---

## Términos y condiciones

Los permisos para utilizar estas publicaciones se otorgan sujetos a los siguientes términos y condiciones.

**Uso personal:** Puede reproducir estas publicaciones para su uso personal, no comercial, siempre y cuando se mantengan los avisos sobre la propiedad. No puede distribuir, visualizar o realizar trabajos derivados de estas publicaciones, o de partes de las mismas, sin el consentimiento expreso de IBM.

**Uso comercial:** Puede reproducir, distribuir y visualizar estas publicaciones únicamente dentro de su empresa, siempre y cuando se mantengan todos los avisos sobre la propiedad. No puede realizar trabajos derivados de estas publicaciones, ni reproducirlas, distribuirlas o visualizarlas, ni de partes de las mismas fuera de su empresa, sin el consentimiento expreso de IBM.

Excepto lo expresamente concedido en este permiso, no se conceden otros permisos, licencias ni derechos, explícitos o implícitos, sobre las publicaciones ni sobre ninguna información, datos, software u otra propiedad intelectual contenida en el mismo.

IBM se reserva el derecho de retirar los permisos aquí concedidos cuando, a su discreción, el uso de las publicaciones sea en detrimento de su interés o cuando, según determine IBM, las instrucciones anteriores no se cumplan correctamente.

No puede descargar, exportar ni volver a exportar esta información excepto en el caso de cumplimiento total con todas las leyes y regulaciones vigentes, incluyendo todas las leyes y regulaciones sobre exportación de los Estados Unidos.

IBM NO GARANTIZA EL CONTENIDO DE ESTAS PUBLICACIONES. LAS PUBLICACIONES SE PROPORCIONAN "TAL CUAL" Y SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUYENDO PERO SIN LIMITARSE A LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN, NO VULNERACIÓN E IDONEIDAD PARA UN FIN DETERMINADO.



---

## Apéndice B. Avisos

Esta información ha sido desarrollada para productos y servicios que se ofrecen en Estados Unidos de América. La información acerca de productos que no son IBM se basa en la información disponible cuando se publicó este documento por primera vez y está sujeta a cambio.

Es posible que IBM no comercialice en otros países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
EE.UU.

Para realizar consultas sobre licencias referentes a información de juegos de caracteres de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país o escribir a:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 Japón

**El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde tales disposiciones sean incompatibles con la legislación local:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede

efectuar, en cualquier momento y sin previo aviso, mejoras y cambios en los productos y programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos sitios web. La información de esos sitios web no forma parte de la información del presente producto de IBM y la utilización de esos sitios web se realiza bajo la responsabilidad del usuario.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited  
Office of the Lab Director  
8200 Warden Avenue  
Markham, Ontario  
L6G 1C7  
CANADÁ

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos el pago de una tarifa.

El programa bajo licencia descrito en este documento y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia de IBM o cualquier acuerdo equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni ninguna otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan objetivos.

Este manual puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos



estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

#### LICENCIA DE COPYRIGHT:

Este manual contiene programas de aplicaciones de ejemplo escritos en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin pago alguno a IBM con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas. Los programas de ejemplo se proporcionan "TAL CUAL", sin ningún tipo de garantía. IBM no se hará responsable de los daños derivados de la utilización que haga el usuario de los programas de ejemplo.

Cada copia o parte de estos programas de ejemplo o cualquier trabajo derivado debe incluir una nota de copyright como la siguiente:

© (*nombre de la empresa*) (*año*). Partes de este código proceden de programas de ejemplo de IBM Corp. © Copyright IBM Corp. *\_entre el o los años\_*. Reservados todos los derechos.

#### **Marcas registradas**

IBM, el logotipo de IBM e [ibm.com](http://ibm.com) son marcas registradas de International Business Machines Corp., que se han registrado en muchas otras jurisdicciones. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Puede consultarse en línea una lista actualizada de las marcas registradas de IBM en la sección Copyright and trademark information de la web [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Los siguientes términos son marcas registradas de otras empresas.

- Linux es una marca registrada de Linus Torvalds en los Estados Unidos y/o en otros países.
- Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.
- UNIX es una marca registrada de The Open Group en los Estados Unidos y/o en otros países.
- Intel, el logotipo de Intel, Intel Inside<sup>®</sup>, el logotipo de Intel Inside, Intel<sup>®</sup> Centrino<sup>®</sup>, el logotipo de Intel Centrino, Celeron<sup>®</sup>, Intel<sup>®</sup> Xeon<sup>®</sup>, Intel SpeedStep<sup>®</sup>, Itanium<sup>®</sup> y Pentium<sup>®</sup> son marcas registradas de Intel Corporation o de sus empresas subsidiarias en Estados Unidos y en otros países.
- Microsoft<sup>®</sup>, Windows, Windows NT<sup>®</sup> y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.



---

# Índice

## A

- actualización de la base de datos
  - Spatial Extender 104
- actualizaciones
  - Centro de información de DB2 510, 512
- actualizar Spatial Extender 27
  - de un sistema de 32 bits a uno de 64 bits 28
- ajuste de índices reticulares espaciales
  - con Index Advisor 85
- anillos
  - definición de regiones geodésicas 137
  - descripción 11
- anotaciones cronológicas
  - diagnóstico 130
- anotaciones cronológicas de administración 130
- aplicaciones
  - programa de ejemplo 113
- ArcExplorer
  - uso como interfaz 95
- atributos ST\_Geometry
  - diferencias geodésicas 169
- avisos 517
- ayuda
  - idioma de configuración 510
  - sentencias SQL 509

## B

- bases de datos
  - habilitación para operaciones espaciales
  - visión general 33

## C

- cadenas lineales 9
- casos prácticos
  - configuración de Spatial Extender 15
- Centro de control
  - mensajes 128
- Centro de información de DB2
  - actualización 510, 512
  - idiomas 510
  - versiones 509
  - visualización en distintos idiomas 510
- cinturón ecuatorial
  - polígonos que lo representan 169
- círculo delimitador mínimo (MBC)
  - atributos ST\_Geometry 169
  - definición 147
  - resultados de funciones espaciales 179
- columnas espaciales
  - geocodificación 68
- comportamiento geodésico
  - ST\_Area 314
  - ST\_Buffer 324
  - ST\_Contains 329
  - ST\_Difference 335
  - ST\_Distance 340
  - ST\_DistanceToPoint 297, 343
  - ST\_Generalize 357

- comportamiento geodésico (*continuación*)
  - ST\_Intersection 374
  - ST\_Intersects 375
  - ST\_Length 385
  - ST\_Perimeter 429
  - ST\_PointAtDistance 301, 436
  - ST\_SymDifference 452
  - ST\_Union 465
  - ST\_Within 467
- consultas
  - funciones espaciales para realizar 95
  - índices espaciales, explotación 96
  - interfaces para someter 95
- conversiones
  - datos espaciales entre sistemas de coordenadas 304
  - mejora de las coordenadas de proceso 50, 54
- coordenadas
  - conversión en sistema de referencia espacial 45
  - conversiones para mejorar el rendimiento 50, 54
  - obtención 285
  - sistemas de referencia espacial 45

## D

- datos de referencia
  - DB2 Spatial Extender
    - descripción 34
- datos espaciales
  - columnas 59
  - descripción 1
  - exportación 65
  - geocodificación 68
  - importación 65
  - recuperación y análisis
    - explotación de índices 96
    - funciones 95
    - interfaces 95
  - ST\_GEOMETRY\_COLUMNS 249, 252
  - tipos de datos 59
  - transferencia de cliente a servidor 477
  - uso 6
- DB2 Geodetic Data Management Feature
  - funciones espaciales con soporte 179
- db2se
  - migrate, mandato 106
  - upgrade, mandato 104
- db2se, mandatos 99
- DB2SE\_USA\_GEOCODER
  - datos de referencia 35
- DE\_HDN\_SRS\_1004
  - sistema de referencia espacial 48
- DEFAULT\_SRS
  - sistema de referencia espacial 48
- densidad de la población mundial
  - estructura de celdas de Voronoi 147
- determinación de problemas
  - guías de aprendizaje 514
  - información disponible 514
- distancia
  - a lo largo de la geodésica 136
  - función ST\_Distance 340

- distancia (*continuación*)
  - ST\_DistanceToPoint 297, 343
  - ST\_PointAtDistance 301, 436
- documentación
  - copia impresa 505
  - PDF 505
  - términos y condiciones de uso 515
  - visión general 505

## E

- ecuador 135
- ejemplos
  - Spatial Extender 113
- elementos geográficos
  - descripción 1
- elipsoides
  - Geodetic Extender 191
- esferoides
  - definición 134
  - en definición de sistema de coordenadas 191
  - sistemas de coordenadas 491
- estructuras de celdas de Voronoi
  - selección de una alternativa para índice 149
- Estructuras de celdas de Voronoi
  - descripción 147
- extensión espacial
  - definición 45

## F

- factores, conversión
  - coordenadas 50, 54
- factores de escala
  - visión general 50, 54
- formación teselada Voronoi 147
- formatos de datos
  - Geography Markup Language (GML) 490
  - representación binaria convencional (WKB) 488
  - representación de forma 490
  - representación de texto convencional (WKT) 483
- fórmulas utilizadas en la geocodificación 50, 54
- función agregada
  - columnas espaciales 311, 475
- funciones
  - espaciales
    - conversiones de formatos de intercambio de datos 263
    - visión general 263
- funciones de agregado de unión 475
- funciones de comparación
  - envolturas de geometrías 282
  - geometrías idénticas 282
  - intersecciones entre geometrías 277, 284
  - relaciones de contenedor 274
  - serie de la matriz patrón DE-9IM 285
  - visión general 271
- funciones de constructor
  - representación binaria convencional 269
  - representación de forma ESRI 270
  - representación de texto convencional 267
  - representación GML (Geography Markup Language) 270
  - visión general 263
- funciones espaciales
  - comparaciones de geometrías
    - envolturas de geometrías 282
    - geometrías idénticas 282

- funciones espaciales (*continuación*)
  - comparaciones de geometrías (*continuación*)
    - intersecciones 277, 284
    - relaciones de contenedor 274
    - serie de la matriz patrón DE-9IM 285
    - visión general 271
  - consideraciones 305
  - conversión de geometrías 263
  - conversiones de datos entre sistemas de coordenadas 304
  - conversiones de formatos de intercambio de datos
    - representación binaria convencional 269
    - representación de forma ESRI 270
    - representación de texto convencional 267
    - representación GML (Geography Markup Language) 270
    - visión general 263
  - diferencias geodésicas en cuanto al comportamiento 179
  - ejemplos 95
  - EnvelopesIntersect 309
  - generación de nuevas geometrías
    - basadas en medidas existentes 297
    - conversión de una en otra 292
    - formas modificadas 302
    - nuevas configuraciones de espacio 293
    - una de varias 296
    - visión general 291
  - información sobre distancia 304
  - información sobre índices 304
  - MBR aggregate (Agregado MBR) 311
  - propiedades de geometrías 285
    - geometrías dentro de una geometría 287
    - información de tipo de datos 285
    - información dimensional 290
    - información sobre configuración 290
    - información sobre medidas y coordenadas 285
    - información sobre perímetros 289
    - sistema de referencia espacial 291
  - que utilizan índices Voronoi geodésicos 147, 150
  - ST\_AppendPoint 313
  - ST\_Area 314
  - ST\_AsBinary 317
  - ST\_AsGML 319
  - ST\_AsShape 320
  - ST\_AsText 321
  - ST\_Boundary 322
  - ST\_Buffer 324
  - ST\_Centroid 327
  - ST\_ChangePoint 328
  - ST\_Contains 329
  - ST\_ConvexHull 331
  - ST\_CoordDim 332
  - ST\_Crosses 334
  - ST\_Difference 335
  - ST\_Dimension 337
  - ST\_Disjoint 338
  - ST\_Distance 340
  - ST\_DistanceToPoint 297, 343
  - ST\_Edge\_GC\_USA 344
  - ST\_Endpoint 348
  - ST\_Envelope 349
  - ST\_EnvIntersects 350
  - ST\_EqualCoordsys 351
  - ST\_Equals 352
  - ST\_EqualSRS 354
  - ST\_ExteriorRing 355
  - ST\_FindMeasure
    - ST\_LocateAlong 298, 356

funciones espaciales (*continuación*)

ST\_Generalize 357  
 ST\_GeomCollection 359  
 ST\_GeomCollFromTxt 361  
 ST\_GeomCollFromWKB 363  
 ST\_Geometry 364  
 ST\_GeometryN 366  
 ST\_GeometryType 367  
 ST\_GeomFromText 368  
 ST\_GeomFromWKB 369  
 ST\_GetIndexParams 370  
 ST\_InteriorRingN 373  
 ST\_Intersection 374  
 ST\_Intersects 375  
 ST\_Is3d 377  
 ST\_IsClosed 378  
 ST\_IsEmpty 380  
 ST\_IsMeasured 381  
 ST\_IsRing 382  
 ST\_IsSimple 383  
 ST\_IsValid 384  
 ST\_Length 385  
 ST\_LineFromText 387  
 ST\_LineFromWKB 388  
 ST\_LineString 389  
 ST\_LineStringN 390  
 ST\_LocateAlong  
     ST\_FindMeasure 298, 356  
 ST\_LocateBetween  
     ST\_MeasureBetween 300, 402  
 ST\_M 392  
 ST\_MaxM 393  
 ST\_MaxX 394  
 ST\_MaxY 396  
 ST\_MaxZ 398  
 ST\_MBR 399  
 ST\_MBRIntersects 400  
 ST\_MeasureBetween  
     ST\_LocateBetween 300, 402  
 ST\_MidPoint 403  
 ST\_MinM 404  
 ST\_MinX 406  
 ST\_MinY 407  
 ST\_MinZ 408  
 ST\_MLineFromText 410  
 ST\_MLineFromWKB 411  
 ST\_MPointFromText 413  
 ST\_MPointFromWKB 414  
 ST\_MPolyFromText 415  
 ST\_MPolyFromWKB 416  
 ST\_MultiLineString 418  
 ST\_MultiPoint 420  
 ST\_MultiPolygon 421  
 ST\_NumGeometries 423  
 ST\_NumInteriorRing 423  
 ST\_NumLineStrings 424  
 ST\_NumPoints 425  
 ST\_NumPolygons 426  
 ST\_Overlaps 427  
 ST\_Perimeter 429  
 ST\_PerpPoints 431  
 ST\_Point 433  
 ST\_PointAtDistance 301, 436  
 ST\_PointFromText 437  
 ST\_PointFromWKB 438  
 ST\_PointN 439  
 ST\_PointOnSurface 440

funciones espaciales (*continuación*)

ST\_PolyFromText 441  
 ST\_PolyFromWKB 442  
 ST\_Polygon 443  
 ST\_PolygonN 445  
 ST\_Relate 446  
 ST\_RemovePoint 448  
 ST\_SRID  
     ST\_SrsId 449  
 ST\_SrsID  
     ST\_SRID 449  
 ST\_SrsName 450  
 ST\_StartPoint 451  
 ST\_SymDifference 452  
 ST\_ToGeomColl 454  
 ST\_ToLineString 456  
 ST\_ToMultiLine 457  
 ST\_ToMultiPoint 458  
 ST\_ToMultiPolygon 459  
 ST\_ToPoint 460  
 ST\_ToPolygon 461  
 ST\_Touches 462  
 ST\_Transform 463  
 ST\_Union 465  
 ST\_Within 467  
 ST\_WKBTToSQL 469  
 ST\_WKTTToSQL 470  
 ST\_X 471  
 ST\_Y 472  
 ST\_Z 473  
 tipos de datos asociados 305  
 Union aggregate (Agregado de unión) 475  
 uso para explotación de índices espaciales 96  
 visión general 263

## G

GCS\_NORTH\_AMERICAN\_1927  
     sistema de coordenadas 48  
 GCS\_NORTH\_AMERICAN\_1983  
     sistema de coordenadas 48  
 GCS\_WGS\_1984  
     sistema de coordenadas 48  
 GCSW\_DEUTSCHE\_HAUPTDRE IECKSNETZ  
     sistema de coordenadas 48  
 geocodificación  
     visión general 68  
 geocodificación automática 68  
 geocodificación en modalidad de proceso por lotes 68  
 geocodificadores  
     configuración de DB2SE\_USA\_GEOCODER 35  
     registro 35  
     ST\_GEOCODER\_PARAMETERS, vista de catálogo 253  
     ST\_GEOCODERS, vista de catálogo 254  
     ST\_GEOCODING\_PARAMETERS, vista de catálogo 256  
     ST\_GEOCODING, vista de catálogo 255  
     ST\_SIZINGS, vista de catálogo 257  
     visión general 68  
 Geodesia 133  
 geodésico  
     definición 136  
     ejemplo 169  
 Geodetic Data Management Feature  
     cuándo se utiliza 133  
     descripción 133  
     elipsoides 191  
     vistas de catálogo espaciales con soporte 184

- Geodetic Extender
  - atributos ST\_Geometry 169
  - diferencias 169
  - procedimientos almacenados espaciales con soporte 184
- Geographic Markup Language (GML), formato de datos 490
- geometrías
  - datos espaciales 6
  - generación de nuevas
    - basadas en medidas existentes 297
    - conversión de una en otra 292
    - formas modificadas 302
    - nuevas configuraciones de espacio 293
    - una de varias 296
    - visión general 291
  - propiedades
    - Véase también "Funciones espaciales, propiedades de geometrías" 285
    - visión general 11
  - transferencia de datos cliente-servidor 477
  - visión general 9
- grados
  - latitud y longitud 135
- grupos de transformación
  - visión general 477
- gse\_disable\_autogc, procedimiento almacenado 208
- gse\_disable\_db, procedimiento almacenado 210
- gse\_disable\_sref, procedimiento almacenado 213
- gse\_enable\_autogc, procedimiento almacenado 214
- gse\_enable\_db, procedimiento almacenado 216
- gse\_enable\_sref, procedimiento almacenado 202
- gse\_export\_shape 218
- gse\_import\_shape, procedimiento almacenado 221
- gse\_register\_gc, procedimiento almacenado 229
- gse\_register\_layer, procedimiento almacenado 234
- gse\_run\_gc, procedimiento almacenado 237
- gse\_unregist\_gc, procedimiento almacenado 244
- gse\_unregist\_layer, procedimiento almacenado 245
- guías de aprendizaje
  - determinación de problemas 514
  - resolución de problemas 514
  - Visual Explain 514

## H

- habilitación
  - operaciones espaciales 33
- hemisferios, polígonos que los representan 169

## I

- ID de sistema de referencia espacial geodésico
  - ST\_create\_srs 202
- identificador de sistemas de referencia espacial (SRID)
  - para datos geodésicos 133, 134
- Index Advisor
  - cuándo se utiliza 78
  - mandato GET GEOMETRY que se debe invocar 91
  - propósito 75, 85
- índice reticular espacial
  - funciones espaciales que los utilizan 84
  - mandato de Index Advisor 91
  - sentencia CREATE INDEX 84
  - sentencias de SQL que los utilizan 84
- índices
  - índices reticulares espaciales
    - descripción 75

- índices (*continuación*)
  - índices reticulares espaciales (*continuación*)
    - sentencia CREATE INDEX 84
  - mandato de Index Advisor 91
  - Voronoi geodésicos
    - estructura de celdas 149
    - sentencia CREATE INDEX 150
- índices espaciales
  - tipos 75
  - Voronoi geodésicos 147
- índices reticulares
  - ajuste 85
  - visión general 75
- índices reticulares espaciales
  - comparados con índices Voronoi geodésicos 75
  - explotación 96
  - niveles y tamaños de retícula 75, 78
- índices Voronoi geodésicos
  - comparados con índices reticulares espaciales 75
  - explotación 96
  - funciones que los explotan 147
  - selección de una estructura Voronoi alternativa 149
  - sentencia CREATE INDEX 150
- información de tipo de datos, obtención 285
- información sobre distancia para geometrías 304
- información sobre índices para geometrías 304
- información sobre medidas, obtención 285
- instalación
  - DB2 Spatial Extender
    - requisitos de hardware y software 22
- interfaces
  - DB2 Spatial Extender 15

## L

- latitud, geodésica
  - definición 135
- latitud geodésica 135
- longitud, geodésica
  - definición 135
- longitud geodésica 135

## M

- Mandato db2se restore\_indexes 107
- Mandato db2se save\_indexes 108
- mandato GET GEOMETRY
  - sintaxis 91
- mandatos
  - db2se 99
- mandatos de Spatial Extender
  - db2se migrate 106
  - db2se restore\_indexes 107
  - db2se save\_indexes 108
  - db2se upgrade 104
- manuales
  - copia impresa
    - pedido 508
- mensajes
  - Centro de control 128
  - funciones 125
  - información sobre formas 126
  - información sobre migración 126
- Spatial Extender
  - CLP 126
  - partes de 121

- mensajes (*continuación*)
  - Spatial Extender (*continuación*)
    - procedimientos almacenados 123
- mensajes de funciones 125
- meridiano 135
- meridiano 180°
  - círculos delimitadores mínimos que lo cruzan 179
  - geometrías que lo cruzan 169
- meridiano 180°, líneas que lo cruzan 169
- meridiano de origen 135
- meridianos de origen
  - sistemas de coordenadas 491
- migración de la base de datos
  - Spatial Extender 106
- multilíneas, colección homogénea de Spatial Extender 9
- multiplicadores para mejorar el rendimiento
  - coordenadas de proceso 50, 54
- multipolígonos, colección homogénea de Spatial Extender 9
- multipuntos, colección homogénea de Spatial Extender 9

## N

- NAD27\_SRS\_1002 (sistema de referencia espacial) 48
- NAD83\_SRS\_1 (sistema de referencia espacial) 48

## P

- pedido de manuales de DB2 508
- polígonos
  - definición de regiones geodésicas 137
  - tipo de geometría 9
- polígonos geodésicos 137
- polos
  - polígonos que los delimitan 169
- procedimientos almacenados
  - problemas 123
  - ST\_alter\_coordsys 194
  - ST\_alter\_srs 196
  - ST\_create\_coordsys 200
  - ST\_create\_srs 202
  - ST\_disable\_autogeocoding 208
  - ST\_disable\_db 210
  - ST\_drop\_coordsys 212
  - ST\_drop\_srs 213
  - ST\_enable\_autogeocoding 214
  - ST\_enable\_db 216
  - ST\_export\_shape 218
  - ST\_import\_shape 221
  - ST\_register\_geocoder 229
  - ST\_register\_spatial\_column 234
  - ST\_remove\_geocoding\_setup 235
  - ST\_run\_geocoding 237
  - ST\_setup\_geocoding 240
  - ST\_unregister\_geocoder 244
  - ST\_unregister\_spatial\_column 245
- procedimientos almacenados espaciales
  - soportadas por Geodetic Data Management Feature 184
- procesador de línea de mandatos (CLP)
  - mandatos de Spatial Extender 99
  - mensajes 126
- propiedades de geometrías
  - funciones espaciales para 285
    - geometrías dentro de una geometría 287
    - información de tipo de datos 285
    - información dimensional 290
    - información sobre configuración 290

- propiedades de geometrías (*continuación*)
  - funciones espaciales para (*continuación*)
    - información sobre medidas y coordenadas 285
    - información sobre perímetros 289
    - sistema de referencia espacial 291
  - visión general 11
- proyecciones azimutales 43
- proyecciones cartográficas
  - sistemas de coordenadas 491
- proyecciones conformes 43
- proyecciones de dirección verdadera 43
- proyecciones de igual área 43
- proyecciones equidistantes 43
- puntos 9

## R

- rectángulo delimitador mínimo (MBR)
  - definición 11
  - en índices reticulares espaciales 75
- regiones geodésicas
  - descripción 137
- registro
  - geocodificadores 35
- rendimiento
  - conversiones de datos de coordenadas 50, 54
  - representación binaria convencional (WKB), formato de datos 488
  - representación de forma, formato de datos 490
  - representación de texto convencional (WKT), formato de datos 483
- requisitos de hardware
  - Spatial Extender 22
- requisitos de software
  - Spatial Extender 22
- resolución de problemas
  - anotaciones cronológicas de administración 130
  - funciones 125
  - guías de aprendizaje 514
  - información en línea 514
  - mensajes de información sobre formas 126
  - mensajes sobre migración 126
  - Spatial Extender
    - mensajes 121
    - procedimientos almacenados 123

## S

- sentencia CREATE INDEX
  - índice reticular espacial 84
  - índice Voronoi geodésico 150
- sentencias de SQL
  - que utilizan índices Voronoi geodésicos 150
- sentencias SQL
  - visualización de la ayuda 509
- sistema de coordenadas geográficas 37
- sistema de coordenadas proyectadas 37
- sistema de referencia
  - en definición de sistema de coordenadas 191
  - geodésico 133, 134
- sistema de referencia de coordenadas
  - latitud y longitud 133
- sistema de referencia espacial (SRS) geodésico
  - descripción 45
- sistema de referencia geodésico 134

- sistemas de coordenadas
  - con soporte 491
  - ST\_COORDINATE\_SYSTEMS, vista de catálogo 251
  - ST\_SPATIAL\_REFERENCE\_SYSTEMS, vista de catálogo 258
  - visión general 37
- sistemas de coordenadas proyectadas 43
- sistemas de referencia espacial
  - creación 202
  - descripción 45
  - suministrado con DB2 Spatial Extender 48
- sistemas de referencia espacial geodésicos 133
- sistemas de referencia geodésicos
  - descripción 133
  - sistemas de coordenadas 491
  - ST\_SPATIAL\_REFERENCE\_SYSTEMS 258
- Spatial Extender
  - actualización, visión general 27
  - actualización desde sistemas de 32 bits a sistemas de 64 bits 28
  - actualizar servidor 27
  - cuándo se utiliza 133
  - datos de referencia 34
  - sistemas de referencia espacial suministrados con 48
- ST\_alter\_coordsys, procedimiento almacenado 194
- ST\_alter\_srs 196
- ST\_COORDINATE\_SYSTEMS 251
- ST\_create\_coordsys, procedimiento almacenado 200
- ST\_create\_srs 202
- ST\_disable\_autogeocoding 208
- ST\_disable\_db, procedimiento almacenado 210
- ST\_Distance 340
- ST\_DistanceToPoint 297, 343
- ST\_drop\_coordsys, procedimiento almacenado 212
- ST\_drop\_srs 213
- ST\_enable\_autogeocoding, procedimiento almacenado 214
- ST\_enable\_db, procedimiento almacenado 216
- ST\_export\_shape, procedimiento almacenado 218
- ST\_GEOCODER\_PARAMETERS 253
- ST\_GEOCODERS 254
- ST\_GEOCODING 255
- ST\_GEOCODING\_PARAMETERS 256
- ST\_GEOMETRY\_COLUMNS 249, 252
- ST\_import\_shape, procedimiento almacenado 221
- ST\_PointAtDistance 301, 436
- ST\_register\_geocoder, procedimiento almacenado 229
- ST\_register\_spatial\_column, procedimiento almacenado 234
- ST\_remove\_geocoding\_setup, procedimiento almacenado 235
- ST\_run\_geocoding, procedimiento almacenado 237
- ST\_setup\_geocoding, procedimiento almacenado 240
- ST\_SIZINGS 257
- ST\_SPATIAL\_REFERENCE\_SYSTEMS 258
- ST\_UNITS\_OF\_MEASURE 261
- ST\_UNITS\_OF\_MEASURE, vista de catálogo 261
- ST\_unregister\_geocoder, procedimiento almacenado 244
- ST\_unregister\_spatial\_column, procedimiento almacenado 245

## T

- tareas
  - configuración de Spatial Extender 15
- términos y condiciones
  - uso de publicaciones 515
- tierra completa
  - representación 169

## U

- unidades angulares
  - sistemas de coordenadas 491
- unidades lineales
  - sistemas de coordenadas 491
- unidades para valores de desplazamiento y factores de escala 50, 54

## V

- valores de desplazamiento
  - visión general 50, 54
- vistas de catálogo
  - ST\_COORDINATE\_SYSTEMS 251
  - ST\_GEOCODER\_PARAMETERS 253
  - ST\_GEOCODERS 254
  - ST\_GEOCODING 255
  - ST\_GEOCODING\_PARAMETERS 256
  - ST\_GEOMETRY\_COLUMNS 249, 252
  - ST\_SIZINGS 257
  - ST\_SPATIAL\_REFERENCE\_SYSTEMS 258
  - ST\_UNITS\_OF\_MEASURE 261
- vistas de catálogo espaciales
  - soportadas por Geodetic Data Management Feature 184
- Visual Explain
  - guía de aprendizaje 514

## W

- WGS84\_SRS\_1003
  - sistema de referencia espacial 48







SC11-3925-00



Spine information:

IBM DB2 9.7 para Linux, UNIX y Windows

**Spatial Extender and Geodetic Data Management Feature Guía del usuario y manual de consulta**

