IBM DB2 10.1
para Linux, UNIX y Windows

Spatial Extender Guía del usuario y manual de consulta

SC11-8081-00

IBM DB2 10.1
para Linux, UNIX y Windows

Spatial Extender Guía del usuario y manual de consulta

SC11-8081-00

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 457.

Nota de edición

Este manual es la traducción del original en inglés *IBM DB2 10.1 for Linux, UNIX, and Windows Spatial Extender User's Guide and Reference* (SC27-3894-00).

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 solicitar publicaciones en línea, vaya a IBM Publications Center en http://www.ibm.com/shop/publications/order
- Para encontrar al representante local de IBM que le corresponde, vaya a la sección Worlwide Contacts de IBM Directory en 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, está otorgando a IBM el derecho no exclusivo de utilizar o distribuir la información de cualquier forma que considere adecuada sin incurrir por ello a ninguna obligación para con usted.

© Copyright IBM Corporation 1998, 2012.

Contenido

© Copyright IBM Corp. 1998, 2012

Capítulo 18. Funciones espaciales 233

Capítulo 1. La finalidad de DB2 Spatial Extender

Utilice DB2 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.

© Copyright IBM Corp. 1998, 2012

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 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, 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 en la página 3 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.

De dónde proceden los datos espaciales

Los datos espaciales pueden obtenerse mediante la utilización de varios métodos.

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. 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 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 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 Concourt 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

Se proporciona un resumen de varios conceptos básicos sobre los que están basadas las operaciones de DB2 Spatial Extender, como son los elementos geográficos, la información espacial, los datos espaciales 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. Geometrías

En DB2 Spatial Extender, la definición funcional de geometría es “un modelo de un elemento geográfico.”

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, 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.

© Copyright IBM Corp. 1998, 2012

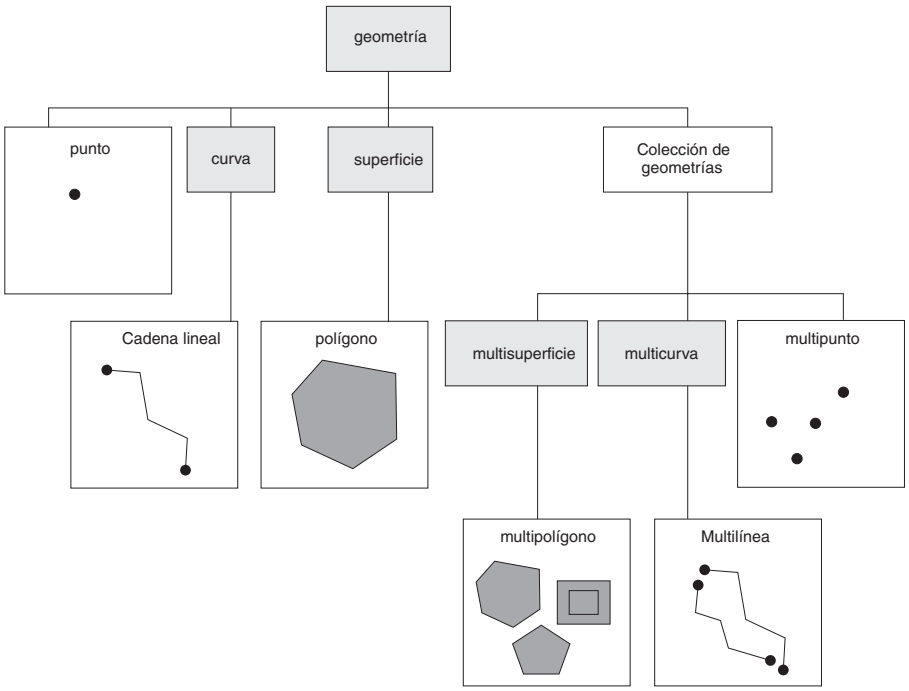


Figura 5. Jerarquía de geometrías soportadas por DB2 Spatial Extender. Las geometrías de esta figura de las que no pueden crearse instancias incluyen ejemplos de cómo se pueden representar visualmente.

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

Tal como indica la 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 que están disponibles para 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

Tipos de geometría

Cada geometría pertenece a un tipo de la jerarquía de geometrías soportadas por DB2 Spatial Extender.

Los tipos de la jerarquía que se va a definir con valores de coordenadas específicos son:
- Puntos
- Cadenas lineales
- Polígonos
- Colecciones de geometrías
- Multipuntos
- Multilíneas
- Multipolígonos

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 Las 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

En el caso de las geometrías que tiene una altitud o profundidad asociada, 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.

Geometrías simples o no simples

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.

Geometrías cerradas

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.

Geometrías vacías o no vacías

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 cuando:

- 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 de las geometrías

Una geometría puede tener un valor de dimensión para indicar si está vacía o si tiene una longitud o un área.

Los valores de dimensión son los siguientes:

–1 Indica que una geometría está vacía.

0 Indica que una geometría carece de longitud y su área es 0 (cero).

1 Indica que una geometría tiene una longitud mayor que 0 (cero) y su área es 0 (cero).

2 Indica que una geometría 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

El soporte y la utilización de DB2 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

Existen gran diversidad de interfaces que pueden utilizarse para configurar DB2 Spatial Extender.

Puede utilizar cualquiera de las interfaces siguientes para configurar DB2 Spatial Extender y crear proyectos que utilicen datos espaciales:

- Programas de aplicación que llaman a procedimientos almacenados de DB2 Spatial Extender.
- Un procesador de línea de mandatos (CLP) proporcionado por DB2 Spatial Extender. Con frecuencia, se hace referencia al CLP de DB2 Spatial Extender como CLP de db2se.
- Consultas SQL que envía desde el CLP de DB2, o desde un programa de aplicación.
- Proyectos de código abierto que incluyen soporte para DB2 Spatial Extender. Algunos proyectos de código abierto con este soporte incluyen:
  - IBM Data Management Geobrowser para DB2 e Informix es un geonavegador para visualizar tablas espaciales y el resultado del análisis espacial en formato gráfico. Para más información, consulte http://www.ibm.com/developerworks/data/tutorials/dm-1103db2geobrowser/index.html
  - GeoTools () es una biblioteca de Java para crear aplicaciones espaciales. Para más información, consulte http://www.geotools.org/
  - GeoServer es un servidor de mapas web y un servidor de funciones web. Para más información, consulte http://geoserver.org/display/GEOS/Welcome
  - uDIG es una aplicación de visualización y análisis de datos espaciales de escritorio. Para más información, consulte http://udig.refractions.net/
  - Productos GIS como Esri ArcGIS

Configuración de DB2 Spatial Extender

En este procedimiento se describen los pasos que debe realizar para configurar DB2 Spatial Extender.

Procedimiento

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).
2. Instale DB2 Spatial Extender.
3. Revise los valores de los parámetros de configuración de base de datos y ajústelos si es necesario. Debe asegurarse de que la base de datos dispone de

© Copyright IBM Corp. 1998, 2012

suficiente memoria y espacio para las funciones espaciales, los archivos de anotaciones cronológicas y las aplicaciones de DB2 Spatial Extender.

4. 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. Para obtener más información, consulte Habilitación de una base de datos para operaciones espacialesHabilitación de una base de datos para operaciones espaciales.

Ejemplo

En el siguiente ejemplo se muestran los pasos que realiza una compañía de seguros de inmuebles ficticia, Safe Harbor Real Estate, para configurar DB2 Spatial Extender:

1. 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. La tabla CUSTOMERS tiene las columnas LATITUDE y LONGITUDE, cuya información se han rellenado como parte del proceso de limpieza de direcciones.

2. El equipo de administración espacial instala DB2 Spatial Extender en un entorno DB2 para UNIX.
3. Un miembro del equipo de administración espacial ajusta las características de los archivos de anotaciones cronológicas de transacción, el tamaño de pila de aplicación y el tamaño de pila de control de aplicaciones de acuerdo con valores adecuados para los requisitos de DB2 Spatial Extender.
4. El equipo de administración espacial configura los recursos que serán necesarios para los proyectos que planean.

- Un miembro del equipo emite un mandato **db2se enable_db** para obtener los recursos que habilitan la base de datos para las operaciones espaciales. Estos recursos incluyen el catálogo de DB2 Spatial Extender, tipos de datos espaciales, funciones espaciales, etcétera.

Qué hacer a continuación

Después de configurar DB2 Spatial Extender, puede empezar a crear proyectos que utilicen datos espaciales.

Creación de proyectos que utilizan datos espaciales

Después de configurar DB2 Spatial Extender, está preparado para acometer proyectos que utilicen datos espaciales. En este procedimiento se describen los pasos que implica la creación de este tipo de proyectos.

Antes de empezar

- Configure DB2 Spatial Extender

Procedimiento

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.)
2. 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.
3. 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.
4. Defina columnas espaciales para el acceso mediante herramientas de visualización, según proceda, 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.

5. Rellene la información de las columnas espaciales realizando una de las acciones siguientes:
   a. Para un proyecto que necesite que se importen datos espaciales, importe los datos.
   b. Para un proyecto que necesite que una columna espacial se establezca a partir de las columnas LATITUDE y LONGITUDE, ejecute una sentencia UPDATE de SQL utilizando el constructor db2gse.ST_Point.
6. 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.
7. Analice la información espacial generada y la información empresarial relacionada. Esta tarea implica consultar columnas espaciales y columnas no espaciales asociadas. En estas consultas, puede incluir funciones de DB2 Spatial Extender que devuelven gran diversidad 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.

Ejemplo

El siguiente ejemplo es una continuación del ejemplo de Configuración de DB2 Spatial Extender. Muestra los pasos que realiza la compañía de seguros de inmuebles Safe Harbor Real Estate para crear un proyecto para la integración de los datos empresariales y espaciales.

1. 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 el sistema de coordenadas GCS_NORTH_AMERICAN_1983 para Estados Unidos que DB2 Spatial Extender proporciona.
   - El equipo de administración espacial decide qué datos se necesitan para cumplir con los objetivos del proyecto y qué tablas contendrán dichos datos.
2. DB2 Spatial Extender proporciona un sistema de referencia espacial, denominado NAD83_SRS_1, que se ha diseñado para utilizarse con GCS_NORTH_AMERICAN_1983. El equipo de administración espacial decide utilizar NAD83_SRS_1.
3. El equipo de administración espacial define columnas para que contengan datos espaciales.
   - El equipo verifica que la tabla ya contiene columnas LATITUDE y LONGITUDE de cliente. Los valores de estas columnas se utilizarán posteriormente para la conversión en valores de puntos espaciales.
   - El equipo crea las tablas OFFICE_LOCATIONS y OFFICE_SALES para que contengan los datos que ahora están almacenados en un sistema de archivos separado utilizando un formato de archivo de formas estándar en el sector.

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.

4. El equipo de administración espacial espera utilizar herramientas de visualización para representar 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.
5. El equipo de administración espacial llena la columna LOCATION en la tabla CUSTOMER, la tabla OFFICE_LOCATIONS, la tabla OFFICE_SALES y una nueva tabla HAZARD_ZONES:
   - El equipo utiliza la sentencia UPDATE CUSTOMERS SET LOCATION = db2gse.ST_Point(LONGITUDE, LATITUDE,1) para rellenar el valor de LOCATION a partir de LATITUDE y LONGITUDE.
   - El equipo importa datos en las tablas OFFICE_LOCATIONS y OFFICE_SALES utilizando el mandato **db2se import_shape**.
   - 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.
6. El equipo de administración espacial crea índices para las columnas registradas. A continuación, crea una vista que une columnas procedentes de las tablas CUSTOMERS y HAZARD ZONES y registra las columnas espaciales en esta vista.
7. El equipo de análisis espacial ejecuta consultas para obtener información que le ayudará a determinar dónde han de establecerse nuevas sucursales y a ajustar las primas en función de la proximidad de los clientes a zonas de riesgo.

Capítulo 4. Iniciación a DB2 Spatial Extender

Familiarícese con las instrucciones para instalar y configurar Spatial Extender en los sistemas operativos soportados. Aprenda también a resolver algunos de los problemas de instalación y configuración que pueden producirse al trabajar con Spatial Extender.

Configuración e instalación de DB2 Spatial Extender

La configuración y la instalación de DB2 Spatial Extender es necesaria para crear un entorno que dé soporte a las operaciones espaciales.

Antes de empezar

Asegúrese de que se cumplen los requisitos de instalación de DB2 Spatial Extender y de los productos de base de datos DB2. Si el sistema no cumple ninguno de los requisitos previos de software, no se realizará la instalación.

Acerca de esta tarea

Un entorno de DB2 Spatial Extender consta de un servidor y un cliente.

Un servidor de DB2 Spatial Extender consta de una instalación de servidor de datos DB2 con el software DB2 Spatial Extender instalado.

Un cliente de DB2 Spatial Extender consta de una instalación de cliente de servidor de datos de IBM® con el software DB2 Spatial Extender instalado.

Las bases de datos habilitadas para las operaciones espaciales se encuentran en el servidor. Se puede acceder a ellas a través de un cliente. 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 en el servidor o en el cliente.

Por lo general, además, un geonavegador forma parte de un componente habitual de un entorno de DB2 Spatial Extender. Aunque no son obligatorios, son útiles para mostrar visualmente los resultados de las consultas espaciales, generalmente en forma de mapas. Con la instalación de DB2 Spatial Extender no se incluye un geonavegador. Sin embargo, puede obtener IBM Data Management Geobrowser para DB2 e Informix para ver los datos espaciales.

Procedimiento

Para configurar e instalar DB2 Spatial Extender en un servidor DB2 o en un cliente de servidor de datos de IBM:

1. Asegúrese de que el sistema cumple con todos los requisitos de software. Consulte “Requisitos del sistema para instalar Spatial Extender” en la página 20.
2. Instale DB2 Spatial Extender; para ello, realice una de las tareas siguientes:
   - “Instalación de DB2 Spatial Extender mediante el asistente de instalación de DB2 (Windows)” en la página 21

© Copyright IBM Corp. 1998, 2012

- "Instalación de DB2 Spatial Extender mediante el asistente de instalación de DB2 (Linux y UNIX)" en la página 22
- Instalación de un producto DB2 utilizando un archivo de respuestas (Windows)
- Instalación de un producto DB2 utilizando un archivo de respuestas (Linux y UNIX)

En el caso de las instalaciones en las que se utiliza un archivo de respuestas, debe indicar las palabras clave siguientes para la instalación del servidor DB2:

```
INSTALL_TYPE=CUSTOM
COMP=SPATIAL_EXTENDER_CLIENT_SUPPORT
COMP=SPATIAL_EXTENDER_SERVER_SUPPORT
```

En el caso de las instalaciones del cliente de servidor de datos de IBM, debe indicar las palabras clave siguientes:

```
INSTALL_TYPE=CUSTOM
COMP=SPATIAL_EXTENDER_CLIENT_SUPPORT
```

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 DB2 Spatial Extender se ha realizado correctamente probando el entorno de DB2 Spatial Extender. Consulte "Verificación de la instalación de Spatial Extender" en la página 24.
5. Opcional: Descargue e instale IBM Data Management Geobrowser para DB2 e Informix. Puede descargarse una copia gratuita desde https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-dm-geobrowser.

Información relacionada

Análisis de datos espaciales de DB2 con un geonavegador gratuito

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.

Sistemas operativos

Puede instalar DB2 Spatial Extender en sistemas operativos de 32 bits en sistemas basados en Intel como:

- Windows
- Linux.

También puede instalar DB2 Spatial Extender en sistemas operativos de 64 bits como:

- AIX
- HP-UX
- Sistema operativo Solaris SPARC
- Linux x86
- Linux para System z
- Windows

Requisitos de software

Para instalar DB2 Spatial Extender, debe cumplir los requisitos de software de los productos de base de datos de DB2. Para obtener más detalles, consulte el apartado "Requisitos de instalación de productos de base de datos DB2" en la publicación *Instalación de servidores DB2*.

Spatial Extender en servidores DB2

Puede instalar Spatial Extender como parte de un producto de base de datos DB2 o bien instalarlo en una copia de DB2 ya existente. Debe cumplir los requisitos de software de los productos de servidor de bases de datos de DB2.

Spatial Extender en los clientes de servidor de datos de IBM

Si instala el cliente de servidor de datos de IBM en sistemas operativos Windows, la instalación personalizada incluye Spatial Extender.

Si instala un producto de servidor de bases de datos DB2 en los sistemas operativos AIX, HP-UX, Solaris, Linux para Intel o Linux en System z, Spatial Extender se puede instalar de forma opcional si se selecciona la instalación personalizada.

Debe cumplir los requisitos de software del cliente de servidor de datos de IBM.

Instalación de DB2 Spatial Extender mediante el asistente de instalación de DB2 (Windows)

Puede utilizar el asistente de instalación de DB2 para instalar DB2 Spatial Extender en los sistemas operativos Windows como parte de una instalación de DB2.

Antes de empezar

Antes de iniciar el asistente de instalación de DB2:

- Asegúrese de que el sistema cumpla los requisitos de instalación, memoria y disco.
- Debe tener una cuenta de usuario local administrador con los derechos de usuario recomendados para llevar a cabo la instalación. En los servidores de bases de datos DB2 donde LocalSystem se pueda utilizar como DAS y usuario de la instancia de DB2 y no se está utilizando la característica de particionamiento de bases de datos, un usuario que no sea administrador pero que tenga privilegios elevados podrá realizar la instalación.

  Nota: Si la instalación del producto se va a realizar mediante una cuenta de usuario no administrador, la biblioteca en tiempo de ejecución VS2010 debe estar instalada antes de intentar instalar un producto de base de datos DB2. Es necesario que la biblioteca en tiempo de ejecución VS2010 se encuentre en el sistema operativo para poder instalar el producto de base de datos DB2. La biblioteca en tiempo de ejecución VS2010 está disponible desde el sitio web de descarga de bibliotecas en tiempo de ejecución de Microsoft. Hay dos opciones: escoja `vcredist_x86.exe` para sistemas de 32 bits o `vcredist_x64.exe` para sistemas de 64 bits.
- Debe cerrar todos los programas para que el programa de instalación pueda actualizar los archivos que corresponda en el sistema sin necesidad de reiniciar.
- En el caso de las instalaciones que se realizan desde una unidad virtual, debe correlacionar la unidad de red con una letra de unidad de Windows. El asistente de instalación de DB2 no da soporte a la instalación desde una unidad virtual ni

desde una unidad de red no correlacionada (como, por ejemplo, *nombresistpral**nombrerecursocompartido* en el Explorador de Windows).

Acerca de esta tarea

Esta tarea forma parte de la tarea más amplia de "Configuración e instalación de DB2 Spatial Extender" en la página 19.

Procedimiento

Para instalar DB2 Spatial Extender como parte de un servidor DB2 o de un cliente de servidor de datos de IBM:

1. Inicie una sesión en el sistema con la cuenta de administrador local que ha indicado para la instalación de DB2.
2. Si dispone del DVD del producto DB2, insértelo en la unidad. Si está habilitada, la función de ejecución automática inicia el Área de ejecución para la instalación de DB2. Si la ejecución automática no funciona, utilice el Explorador de Windows para examinar el DVD del producto de base de datos DB2 y efectúe una doble pulsación sobre el icono de **instalación** para iniciar el Área de ejecución para la instalación de DB2.
3. Si ha descargado el producto de base de datos DB2 de Passport Advantage, ejecute el archivo ejecutable para extraer los archivos de instalación del producto de base de datos DB2. Utilice el Explorador de Windows para examinar los archivos de instalación de DB2 y efectúe una doble pulsación sobre el icono **setup** para iniciar el Área de ejecución para la instalación de DB2.
4. Lea los requisitos previos de instalación y las notas de release del Área de ejecución para la instalación de DB2 o bien inicie directamente la instalación.
5. Pulse **Instalar un producto**; la ventana Instalar un producto muestra los productos disponibles para la instalación.
6. Instale Spatial Extender utilizando uno de los métodos siguientes:
   - Para crear una copia nueva de DB2 con Spatial Extender, pulse **Instalar nuevo** para iniciar la instalación. Seleccione el tipo de instalación personalizada para incluir Spatial Extender como parte de la instalación. Avance por la instalación siguiendo las indicaciones del asistente de instalación de DB2.
   - Para instalar Spatial Extender en una copia de DB2 ya existente, pulse **Trabajar con existente**. Seleccione el tipo de instalación personalizada para incluir Spatial Extender como parte de la instalación. Avance por la instalación siguiendo las indicaciones del asistente de instalación de DB2.

   Una vez finalizada la instalación, compruebe si hay mensajes de aviso o error en el archivo de anotaciones cronológicas identificado.

Qué hacer a continuación

Después de instalar DB2 Spatial Extender, cree una instancia de DB2 en las copias de DB2 nuevas y verifique que la instalación se ha realizado correctamente.

Instalación de DB2 Spatial Extender mediante el asistente de instalación de DB2 (Linux y UNIX)

Para instalar DB2 Spatial Extender en los sistemas operativos Linux o UNIX, utilice el asistente de Instalación de DB2 o un archivo de respuestas.

Antes de empezar

Antes de iniciar el asistente de instalación de DB2:

- Asegúrese de que el sistema cumpla los requisitos de instalación, memoria y disco.
- Asegúrese de que tiene instalado un navegador soportado.
- Asegúrese de que la imagen del producto de base de datos DB2 está disponible en el sistema. Para obtener una imagen de instalación de DB2, puede adquirir un DVD físico del producto de base de datos DB2 o descargar una imagen de instalación de Passport Advantage.
- Si instala una versión no inglesa del producto de base de datos DB2, asegúrese de que también dispone del Paquete de idiomas nacionales adecuado.
- Asegúrese de que ha instalado software de X Linux que pueda presentar un usuario gráfico, de que el servidor X Linux está en ejecución y de que la variable *DISPLAY* está definida. El asistente de instalación de DB2 es un instalador gráfico.
- Si utiliza software de seguridad en el entorno, deberá crear manualmente los usuarios de DB2 necesarios antes de iniciar el asistente de instalación de DB2.

Acerca de esta tarea

Esta tarea forma parte de la tarea más amplia de "Configuración e instalación de DB2 Spatial Extender" en la página 19.

Puede instalar un producto de base de datos DB2 utilizando la autorización de usuario root o no root.

Procedimiento

Para instalar DB2 Spatial Extender como parte de un servidor DB2 o de un cliente de servidor de datos de IBM:

1. Si tiene un DVD físico del producto de base de datos DB2, vaya al directorio en el que está montado el DVD del producto de base de datos DB2 entrando el mandato siguiente:

   ```
   cd /dvdrom
   ```

   donde */dvdrom* representa el punto de montaje del DVD del producto de base de datos DB2.
2. Si ha descargado la imagen del producto de base de datos DB2, deberá extraer y desempaquetar el archivo del producto.
   a. Extraiga el archivo del producto:

      ```
      gzip -d producto.tar.gz
      ```

      donde *producto* es el nombre del producto que ha descargado.
   b. Desempaquete el archivo del producto:

      En sistemas operativos Linux

      ```
      tar -xvf producto.tar
      ```

      En sistemas operativos AIX, HP-UX y Solaris

      ```
      gnutar -xvf producto.tar
      ```

      donde *producto* es el nombre del producto que ha descargado.
   c. Cambie de directorio:

```
cd ./producto
```

donde *producto* es el nombre del producto que ha descargado.

Nota: Si ha descargado un Paquete de idiomas nacionales, desempaquételo en el mismo directorio. Esto creará los subdirectorios (por ejemplo, `./nlpack`) en el mismo directorio y permitirá al programa de instalación hallar las imágenes de instalación de forma automática, sin preguntar.

3. Entre el mandato **./db2setup** desde el directorio donde resida la imagen del producto de base de datos para iniciar el asistente de instalación de DB2.
4. Se abre el Área de ejecución para la instalación de IBM DB2. Desde esta ventana, puede ver los requisitos previos de la instalación y las notas del release, o bien puede ir directamente a la instalación. Para obtener información de última hora, puede repasar los requisitos previos de la instalación y las notas del release.
5. Pulse **Instalar un producto** y la ventana **Instalar un producto** mostrará los productos disponibles para la instalación.
6. Instale Spatial Extender utilizando uno de los métodos siguientes:
   - Para crear una copia nueva de DB2 con Spatial Extender, pulse **Instalar nuevo** para iniciar la instalación. Seleccione el tipo de instalación personalizada para incluir Spatial Extender como parte de la instalación. Avance por la instalación siguiendo las indicaciones del asistente de instalación de DB2.
   - Para instalar Spatial Extender en una copia de DB2 ya existente, pulse **Trabajar con existente**. Seleccione el tipo de instalación personalizada para incluir Spatial Extender como parte de la instalación. Avance por la instalación siguiendo las indicaciones del asistente de instalación de DB2.

   Una vez finalizada la instalación, compruebe si hay mensajes de aviso o error en el archivo de anotaciones cronológicas identificado.

Qué hacer a continuación

Después de instalar DB2 Spatial Extender, cree una instancia de DB2 en las copias de DB2 nuevas y verifique que la instalación se ha realizado correctamente.

Verificación de la instalación de Spatial Extender

Después de haber instalado DB2 Spatial Extender debe validar la instalación.

Antes de empezar

Antes de validar una instalación de DB2 Spatial Extender, deben cumplirse los siguientes requisitos:

- DB2 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 DB2 Spatial Extender. Para más información, consulte “Configuración e instalación de DB2 Spatial Extender” en la página 19 La tarea consiste en crear una base de datos DB2 y ejecutar una aplicación de ejemplo de DB2 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 verificar la instalación de DB2 Spatial Extender:

1. Linux y UNIX: Inicie sesión en el sistema con el ID de usuario que corresponda al rol de propietario de la instancia 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 superior y con un tamaño mínimo de 500 páginas. Si no dispone de un espacio de tablas de este tipo, consulte "Creación de espacios de tablas temporales" en *Database Administration Concepts and Configuration Reference* para obtener más información sobre cómo crearlo. Se trata de un requisito para ejecutar el programa runGSEdemo.
4. Para los sistemas operativos Windows, establezca el parámetro de configuración de base de datos **agent_stack_sz** en el valor 100 o en un valor superior. En el siguiente ejemplo se muestra el mandato de CLP para establecer el parámetro en 110:

   ```
   UPDATE DBM CFG USING AGENT_STACK_SZ 110
   ```
5. Vaya al directorio donde reside el programa runGSEdemo:
   - Para las instalaciones de DB2 Spatial Extender en sistemas operativos Linux y UNIX, especifique:

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

     donde *$HOME* es el directorio inicial del propietario de la instancia.
   - Para la instalación de DB2 Spatial Extender en sistemas operativos Windows, especifique:

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

     donde *c:\Archivos de programa\IBM\sqllib* es el directorio en el que está 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 mibd IDusuario contraseña
   ```

   donde *mi_bd* es el nombre de la base de datos.

Capítulo 5. Actualización a DB2 Spatial Extender Versión 10.1

La actualización a DB2 Spatial Extender para Versión 10.1 en sistemas en los que ha instalado DB2 Spatial Extender Versión 9.5 o Versión 9.7 requiere más tareas que la simple instalación de DB2 Spatial Extender paraVersión 10.1. Debe realizar la tarea de actualización adecuada para estos sistemas.

Las siguientes tareas describen todos los pasos necesarios para actualizar DB2 Spatial Extender desde la Versión 9.5 o la Versión 9.7 hasta Versión 10.1:

- "Actualización de DB2 Spatial Extender"
- "Actualización de DB2 Spatial Extender de 32 bits a 64 bits" en la página 28

Si el entorno DB2 tiene otros componentes, como servidores de DB2, clientes y aplicaciones de base de datos, consulte "Actualización a DB2 Versión 10.1" en el manual *Actualización a DB2 Versión 10.1* para obtener información detallada acerca de la forma de 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 habilitadas para operaciones 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 10.1.
- 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 superior y con un tamaño mínimo de 500 páginas.

Acerca de esta tarea

Si ha instalado DB2 Spatial Extender Versión 9.5 o Versión 9.7, debe realizar los pasos siguientes antes de utilizar una base de datos existente habilitada para operaciones espaciales con DB2 Spatial Extender Versión 10.1. En este tema se describen los pasos necesarios para actualizar bases de datos habilitadas para operaciones espaciales desde una versión anterior de DB2 Spatial Extender.

Procedimiento

Para actualizar DB2 Spatial Extender a Versión 10.1:

1. Actualice el servidor de DB2 desde la Versión 9.5 o la Versión 9.7 hasta Versión 10.1 utilizando una de las tareas siguientes:
   - "Actualización de un servidor de DB2(Windows)" en *Actualización a DB2 Versión 10.1*
   - "Actualización de un servidor de DB2 (Linux y UNIX)" en *Actualización a DB2 Versión 10.1*

© Copyright IBM Corp. 1998, 2012

En la tarea de actualización, debe instalar DB2 Spatial Extender Versión 10.1 después de haber instalado DB2 Versión 10.1 para que la actualización de sus instancias se realice correctamente.

2. Finalice todas las conexiones con la base de datos.
3. Actualice las bases de datos habilitadas para operaciones espaciales desde la Versión 9.5 o la Versión 9.7 hasta la Versión 10.1 utilizando el mandato **db2se upgrade**.

Resultados

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 de 32 bits a 64 bits

La actualización desde DB2 Spatial Extender Versión 10.1 de 32 bits hasta DB2 Spatial Extender Versión 10.1 de 64 bits requiere que realice una copia de seguridad de los índices espaciales, que actualice el servidor de DB2 por DB2 Versión 10.1 de 64 bits, que instale DB2 Spatial Extender Versión 10.1 de 64 bits y que, a continuación, restaure los índices espaciales de los que ha realizado una 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 superior y con un tamaño mínimo de 500 páginas.

Acerca de esta tarea

Si desea actualizar desde DB2 Spatial Extender Versión 9.5 o Versión 9.7 de 32 bits hasta DB2 Spatial Extender Versión 9.7 de 64 bits, en lugar de ello, 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 el servidor de DB2 desde la Versión 9.5 o la Versión 9.7 de 32 bits hasta DB2 Versión 10.1 de 64 bits utilizando una de las tareas siguientes:
   - “Actualización de las instancias de 32 bits de DB2 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 10.1.
5. Restaure los índices espaciales emitiendo el mandato **db2se restore_indexes** desde un indicador de mandatos del sistema operativo.

Capítulo 6. 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.

Los recursos espaciales comprenden:

- Los objetos proporcionados por Spatial Extender para dar soporte a las operaciones espaciales. Por ejemplo, procedimientos almacenados para administrar una base de datos y tipos de datos y programas de utilidad espaciales para geocodificar, importar o exportar datos espaciales.
- Cualquier codificador que los usuarios o los proveedores proporcionen.

Para que estos recursos estén disponibles, debe habilitar la base de datos para las operaciones espaciales, configurar el acceso a los datos de referencia y registrar los geocodificadores de terceros.

Inventario de recursos suministrados para la base de datos

DB2 Spatial Extender proporciona a la base de datos varios recursos para su habilitación con el fin de dar soporte a las operaciones espaciales.

Estos recursos son:

- 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.

© Copyright IBM Corp. 1998, 2012

Habilitación de una base de datos para operaciones espaciales

La habilitación de la base de datos para las operaciones espaciales consiste en establecer DB2 Spatial Extender de modo que proporcione una base de datos con recursos para crear columnas espaciales y manipular datos espaciales.

Antes de empezar

Antes de habilitar una base de datos para operaciones espaciales:

- Asegúrese de que el ID de usuario tiene autorización DBADM en 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 superior y con un tamaño mínimo de 500 páginas.

Procedimiento

Habilite una base de datos para las operaciones espaciales de cualquiera de las formas siguientes:

- Emita el mandato **db2se enable_db**.
- Ejecute una aplicación que llame al procedimiento almacenado DB2GSE.ST_ENABLE_DB.

Puede seleccionar explícitamente el espacio de tablas donde desea que resida el catálogo de DB2 Spatial Extender. Si no lo hace, el sistema de bases de datos de DB2 utilizará el espacio de tablas por omisión.
Si desea utilizar datos espaciales en un entorno de base de datos particionada, no utilice el espacio de tablas por omisión. Para obtener mejores resultados, habilite la base de datos en un espacio de tablas definido para un solo nodo. Por lo general, un espacio de tablas de un solo nodo se define para las tablas pequeñas cuando el particionamiento no es útil.
Por ejemplo, puede definir un espacio de tablas de un solo nodo mediante el procesador de línea de mandatos db2se:

```
db2se enable_db my_db -tableCreationParameters "IN NODE0TBS"
```

Se puede aumentar la eficiencia de muchas consultas si la tabla de sistemas de referencia espacial DB2GSE.GSE_SPATIAL_REFERENCE_SYSTEMS se replica en todos los nodos que se utilizan para las tablas empresariales a las que se emitirán las consultas. Puede volver a crear la tabla DB2GSE.GSE_SRS_REPLICATED_AST con sentencias como la siguiente:

```
drop table db2gse.gse_srs_replicated_ast;
-- MQT para replicar la información de SRS en todos los nodos
-- en un entorno de base de datos particionada
CREATE TABLE db2gse.gse_srs_replicated_ast AS
   ( SELECT srs_name, srs_id, x_offset, x_scale, y_offset, z_offset, z_scale,
       m_offset, m_scale, definition
     FROM db2gse.gse_spatial_reference_systems )
   DATA INITIALLY DEFERRED
   REFRESH IMMEDIATE
   ENABLE QUERY OPTIMIZATION
   REPLICATED
   IN ts_data  espacio_tablas_particionado
;

REFRESH TABLE db2gse.gse_srs_replicated_ast;

CREATE  INDEX db2gse.gse_srs_id_ast
   ON db2gse.gse_srs_replicated_ast ( srs_id )
```

```
;

RUNSTATS ON TABLE db2gse.gse_srs_replicated_ast and indexes all;
```

Registro de un geocodificador

Para poder utilizar geocodificadores, primero deben registrarse.

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:

- Emita el mandato **db2se register_gc**.
- Ejecute una aplicación que llame al procedimiento DB2GSE.ST_REGISTER_GEOCODER.

Capítulo 7. 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.

Es importante que conozca la naturaleza de los sistemas de coordenadas y que sepa qué son los sistemas de referencia espacial.

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 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.

Por ejemplo, en la Figura 6 en la página 34 se muestra un sistema de coordenadas geográficas en el que una ubicación se representa mediante las coordenadas de 80

© Copyright IBM Corp. 1998, 2012

grados este de longitud y de 55 grados norte de latitud.

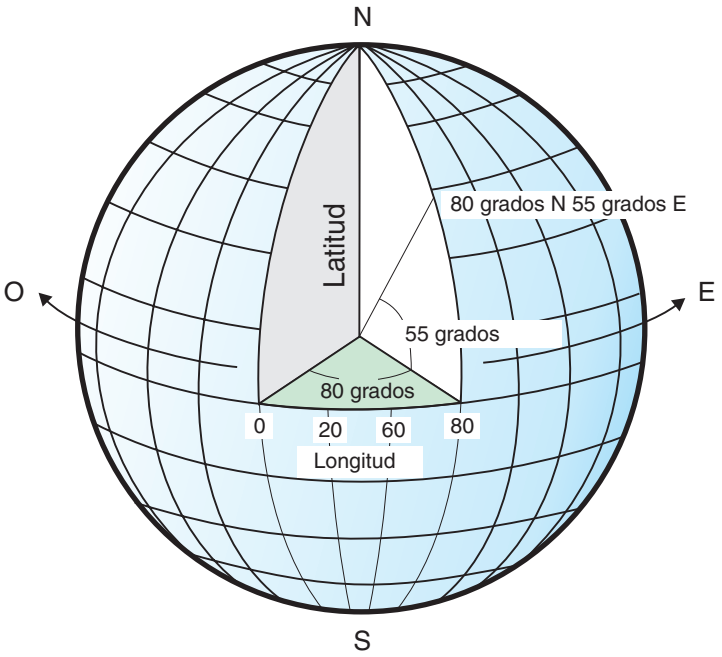


Figura 6. Un sistema de coordenadas geográficas

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 35 ilustra las líneas de latitud.

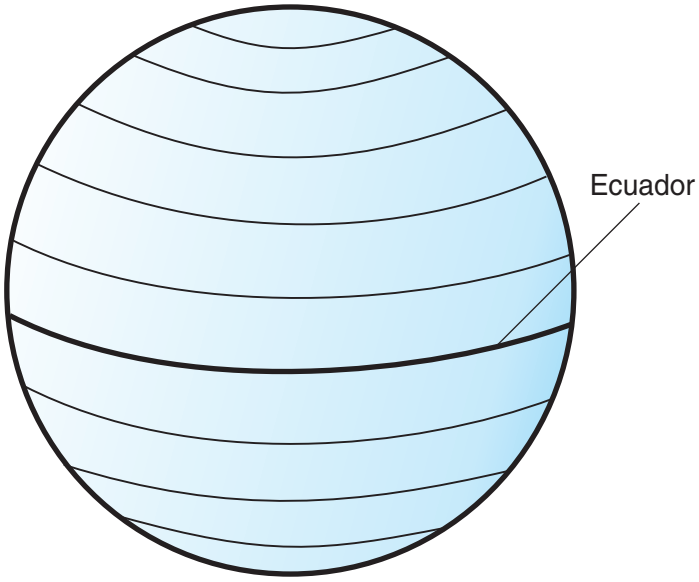


Figura 7. Líneas de latitud

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 36 muestra las líneas de longitud.

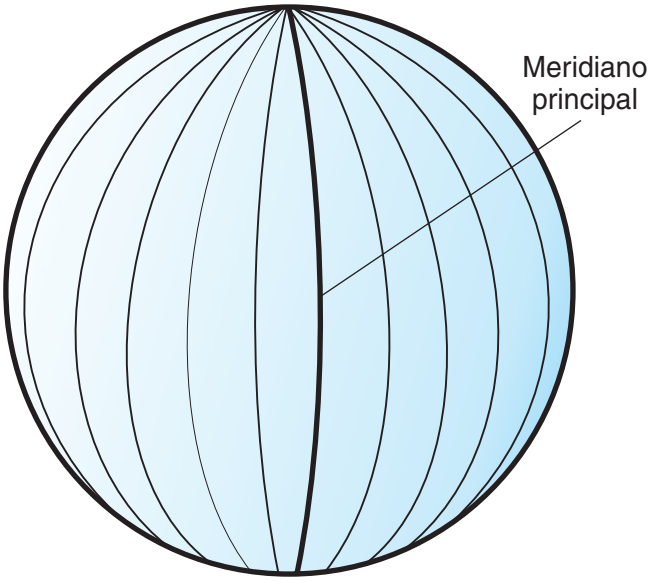


Figura 8. Líneas de longitud

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 37 muestra las diferentes dimensiones entre ubicaciones en la red geográfica.

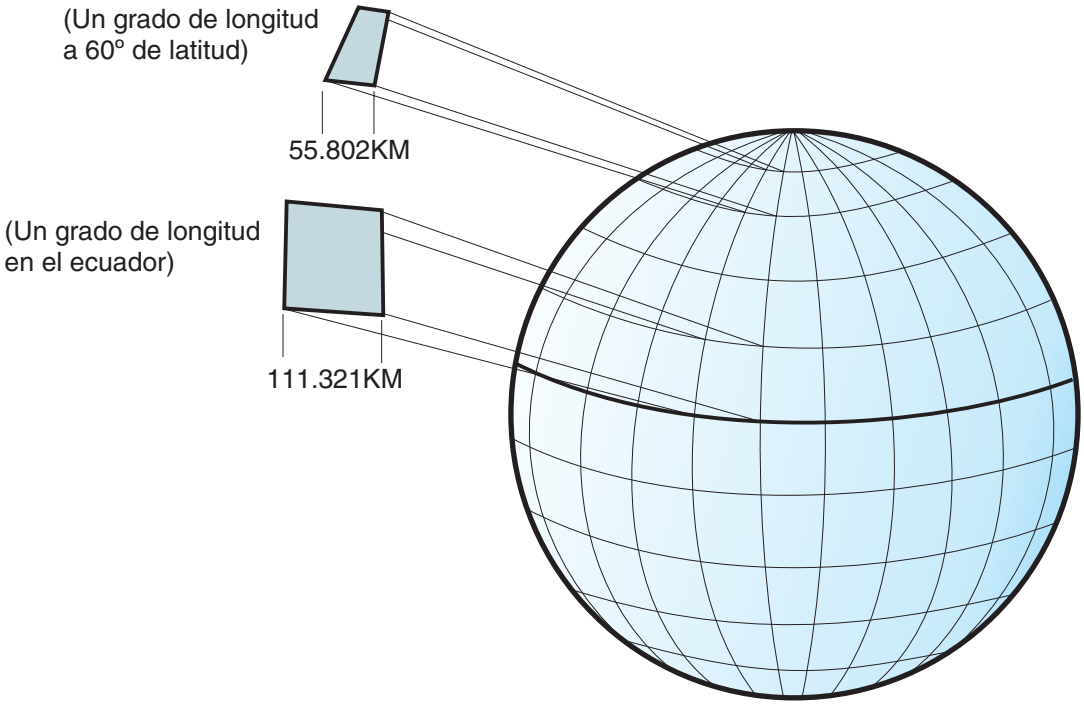


Figura 9. Diferentes dimensiones entre ubicaciones en la red geográfica

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 38 se muestran las aproximaciones de esfera y esferoide de la tierra y los ejes mayor y menor de una elipse.

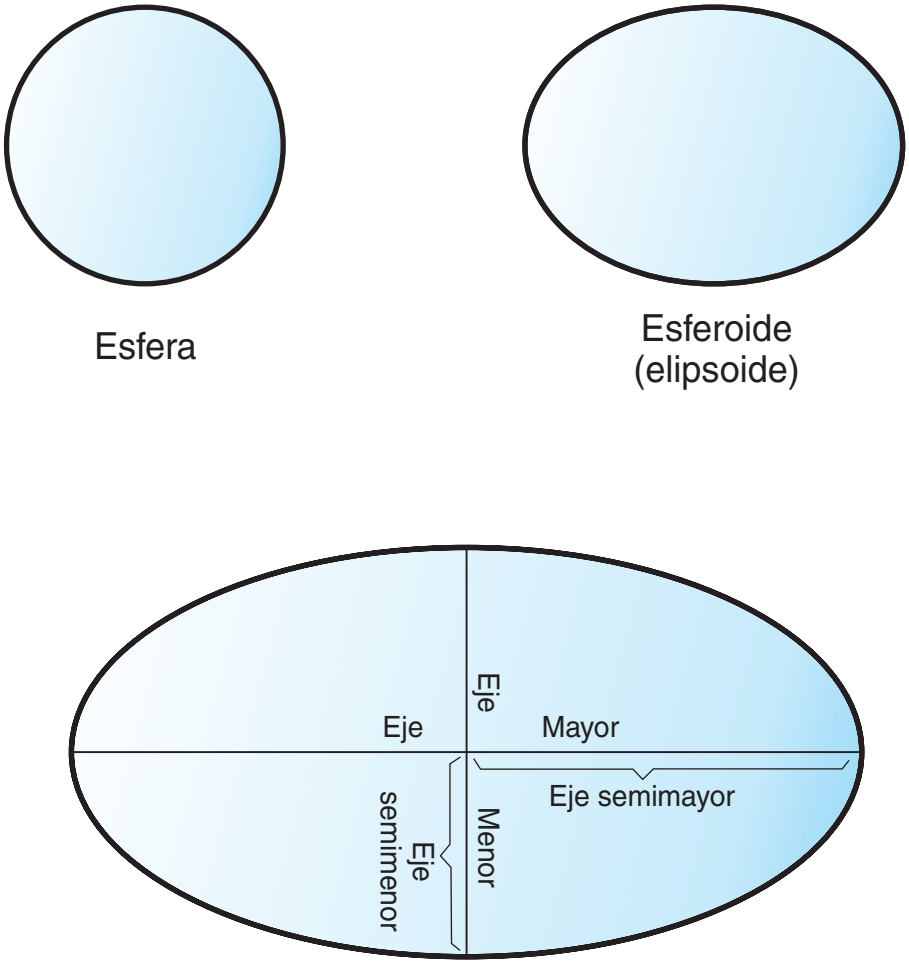


Figura 10. Aproximaciones de esfera y esferoide

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 39 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.

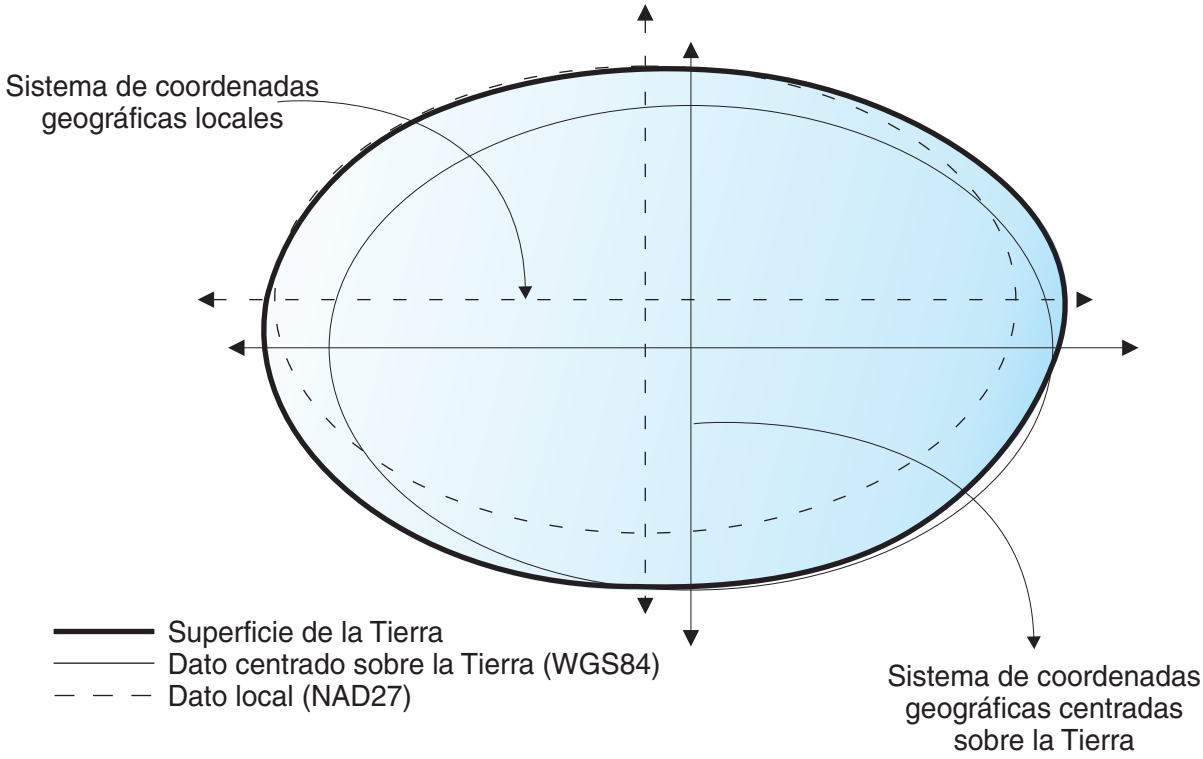


Figura 11. Alineaciones de sistemas de referencia

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.

Determinación del sistema de coordenadas que ha de utilizarse

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

Antes de empezar

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.

Acerca de esta tarea

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.

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.

Procedimiento

Para determinar qué sistema de coordenadas ha de utilizarse:

1. Revise los sistemas de coordenadas existentes que se entregan con DB2 Spatial Extender y utilice el sistema de referencia espacial correspondiente que está basado en el sistema de coordenadas de su elección.

   Los sistemas de coordenadas más utilizados son:

   - 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.
   - 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.
2. Si no encuentra un sistema de coordenadas existente en Spatial Extender, puede crear uno utilizando uno de los métodos siguientes:
   - Emita el mandato db2se create_cs desde el procesador de línea de mandatos db2se.
   - Ejecute una aplicación que llame al procedimiento DB2SE.ST_CREATE_COORDSYS.

   Rara vez es necesario crear un sistema de coordenadas. Asimismo, también necesitará crear un sistema de referencia espacial basado en el nuevo sistema de coordenadas.

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 se utiliza para representar una geometría.

Estos parámetros son:

- 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.

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 finalizar el quinto año, las coordenadas de las planicies situadas más al oeste, este, norte y sur. 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 en función del crecimiento de las planicies aluviales.

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.

Determinación de si ha de utilizarse un sistema de referencia existente o crearse un nuevo sistema

Responder a la siguiente serie de preguntas puede ayudarle a decidir si ha de utilizarse un sistema de referencia espacial existente o crearse un nuevo sistema.

Acerca de esta tarea

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.

Procedimiento

Para decidir si ha de utilizarse un sistema de referencia existente o crearse un nuevo sistema:

1. Responda a las siguientes preguntas para determinar si puede utilizar uno de los sistemas de referencia espacial existentes.
   a. ¿El sistema de coordenadas en el que está basado el sistema de referencia espacial existente 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 44.
   b. ¿Los factores de conversión asociados a uno de los sistemas de referencia espacial existentes funcionan con sus 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 existentes:

1) Revise la información en el apartado "Factores de conversión que transforman datos de coordenadas en enteros" en la página 47.
2) Observe cómo están definidos estos factores para los sistemas de referencia espacial existentes. 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 48.

c. ¿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.

2. Si los sistemas de referencia espacial existentes no funcionan con los datos, deberá acceder al apartado "Creación de un sistema de referencia espacial" en la página 48.

Qué hacer a continuación

Después de decidir qué sistema de referencia espacial necesita, debe especificar su elección para Spatial Extender en funciones o llamadas de procedimiento.

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 catálogoDB2GSE.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 1. Sistemas de referencia espacial proporcionados con DB2 Spatial Extender

Sistema de referencia espacial	ID de SRS	Sistema de coordenadas	Valores de desplaza-miento	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.

Tabla 1. Sistemas de referencia espacial proporcionados con DB2 Spatial Extender (continuación)

Sistema de referencia espacial	ID de SRS	Sistema de coordenadas	Valores de desplaza-miento	Factores de escala	Cuándo se utiliza
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.
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.

Tabla 1. Sistemas de referencia espacial proporcionados con DB2 Spatial Extender (continuación)

Sistema de referencia espacial	ID de SRS	Sistema de coordenadas	Valores de desplaza-miento	Factores de escala	Cuándo se utiliza
DE_HDN _SRS_1004	1004	GCSW _DEUTSCHE _HAUPTDRE IECKSNETZ	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).

$(max(x) - xOffset) * xScale = 2^{53}$
$(max(y) - yOffset) * yScale = 2^{53}$
$(max(z) - zOffset) * zScale = 2^{53}$
$(max(m) - mOffset) * mScale = 2^{53}$

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.

Procedimiento

Para crear un sistema de referencia espacial:

1. Elija un ID de sistema de referencia espacial (SRID) que todavía no esté utilizándose.
2. Decida el grado de precisión del sistema de referencia espacial mediante uno de los métodos siguientes:
   - Indicando 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 calcula los valores de desplazamiento.
   - Indicando 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 deba seguir criterios estrictos para lograr mayor precisión.
3. Calcule la información de conversión que Spatial Extender necesita para convertir datos de coordenadas en enteros positivos. La información que se calcula varía en función del método que haya elegido en el paso 2.
   - Para el método de extensiones, calcule 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, establezca el factor de escala en un número que convierta la parte decimal en un valor entero. Por ejemplo, si las unidades de coordenadas son metros y la precisión de los datos es 1 cm, necesitaría un factor de escala 100.
     – *Valores mínimos y máximos* para las coordenadas y medidas.
   - Para el método de desplazamiento, calcule la información siguiente:
     – *Valores de desplazamiento*. Si los datos de coordenadas incluyen números o medidas negativos, especifique 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. 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.
4. Emita el mandato **db2se create_srs** o llame al procedimiento DB2SE.ST_CREATE_SRS para crear el sistema de referencia espacial.

   En el ejemplo siguiente se muestra cómo crear un sistema de referencia espacial llamado mysrs:

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

Cálculo de los factores de escala

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.

Acerca de esta tarea

Después de calcular los factores de escala deberá determinar los valores de extensión. A continuación, emita el mandato **db2se create_srs** o llame al procedimiento almacenado DB2SE.ST_CREATE_SRS.

Procedimiento

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 coordendada 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

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

Acerca de esta tarea

Una vez determinados los valores de extensión, si alguna de las coordenadas son valores decimales, deberá calcular los factores de escala. En caso contrario, emita el mandato **db2se create_srs** o llame al procedimiento almacenado DB2SE.ST_CREATE_SRS.

Procedimiento

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, especifique valores de desplazamiento. Un desplazamiento es un número que se resta de todas las coordenadas, dejando sólo valores positivos como resto.

Acerca de esta tarea

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. Puede mejorar el rendimiento de las operaciones espaciales si

las coordenadas son enteros positivos en vez de números o medidas negativas.

Procedimiento

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.

Procedimiento

Para crear un sistema de referencia espacial:

1. Elija un ID de sistema de referencia espacial (SRID) que todavía no esté utilizándose.
2. Decida el grado de precisión del sistema de referencia espacial mediante uno de los métodos siguientes:
   - Indicando 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 calcula los valores de desplazamiento.
   - Indicando 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 deba seguir criterios estrictos para lograr mayor precisión.

3. Calcule la información de conversión que Spatial Extender necesita para convertir datos de coordenadas en enteros positivos. La información que se calcula varía en función del método que haya elegido en el paso 2 en la página 49.
   - Para el método de extensiones, calcule 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, establezca el factor de escala en un número que convierta la parte decimal en un valor entero. Por ejemplo, si las unidades de coordenadas son metros y la precisión de los datos es 1 cm, necesitaría un factor de escala 100.
     - *Valores mínimos y máximos* para las coordenadas y medidas.
   - Para el método de desplazamiento, calcule la información siguiente:
     - *Valores de desplazamiento*. Si los datos de coordenadas incluyen números o medidas negativos, especifique 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. 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.
4. Emita el mandato **db2se create_srs** o llame al procedimiento DB2SE.ST_CREATE_SRS para crear el sistema de referencia espacial.

   En el ejemplo siguiente se muestra cómo crear un sistema de referencia espacial llamado mysrs:

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

Capítulo 8. Configuración de columnas espaciales

En la preparación para obtener datos espaciales para un proyecto, no debe seleccionar un sistema de coordenadas y un sistema de referencia espacial y después proporcionar una o varias columnas de tabla donde contener los datos.

Visualización de columnas espaciales

Cuando utiliza una herramienta de visualización, como ArcExplorer for DB2, para consultar una columna espacial, la herramienta devuelve resultados en forma de una representación geográficas; 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 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.

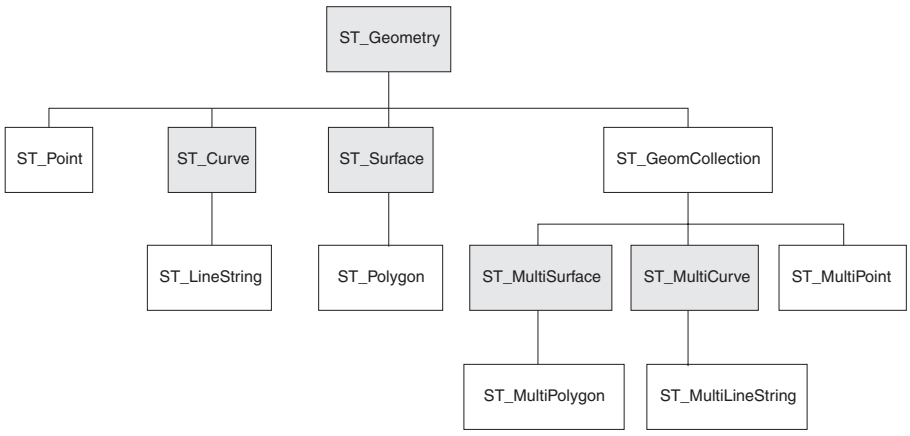


Figura 12. Jerarquía de tipos de datos espaciales. De los tipos de datos que aparecen en los cuadros blancos pueden crearse instancias. De los tipos de datos que aparecen en los cuadros sombreados no pueden crearse instancias.

La jerarquía de la Figura 12 incluye:

© Copyright IBM Corp. 1998, 2012

- 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 varias unidades

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: Determinadas herramientas de visualización no dan soporte a las columnas ST_Geometry, sino que sólo dan soporte a las columnas a las que se ha asignado un subtipo adecuado de ST_Geometry.

Creación de columnas espaciales

Debe crear columnas espaciales para almacenar y recuperar datos espaciales.

Antes de empezar

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 tablas, 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

Acerca de esta tarea

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.

Procedimiento

Para crear columnas espaciales:

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

- Utilice la sentencia CREATE TABLE de SQL para crear una tabla y para incluir una columna espacial dentro de esa tabla.
- Utilice la sentencia ALTER TABLE de SQL para añadir una columna espacial a una tabla existente.
- 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".

Qué hacer a continuación

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.

Antes de empezar

Si utiliza el procesador de línea de mandatos db2se o un programa de aplicación, el ID de usuario debe disponer de autorización SYSADM o DBADM para la base de datos.

Acerca de esta tarea

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.

Procedimiento

Registre una columna espacial utilizando uno de los métodos siguientes:

- Emita el mandato db2se register_spatial_column.
- Ejecute una aplicación que invoque al procedimiento DB2GSE.ST_REGISTER_SPATIAL_COLUMN.

Consulte la columna SRS_NAME de la vista DB2GSE.ST_GEOMETRY_COLUMNS para comprobar el sistema de referencia espacial que ha seleccionado para una columna en particular después de haber registrado la columna.

Capítulo 9. Cómo llenar columnas espaciales

Después de crear columnas espaciales y registrar aquellas a las que accederán estas herramientas de visualización, puede llenar las columnas con datos espaciales importándolos, utilizando un geocodificador para obtenerlos de los datos comerciales o utilizando 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 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:

© Copyright IBM Corp. 1998, 2012

- 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 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 empezar

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

Acerca de esta tarea

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.

Procedimiento

Para importar datos de formas en una tabla nueva o existente:

Elija qué método desea utilizar para importar datos de formas:

- Emita el mandato db2se import_shape.
- Ejecute una aplicación que llame al procedimiento DB2GSE.ST_IMPORT_SHAPE.

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.

Antes de empezar

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.

Acerca de esta tarea

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.

Procedimiento

Para exportar datos a un archivo de formas:

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

- Emita el mandato db2se desde el procesador de línea de mandatos db2se.
- Ejecute una aplicación que llame al procedimiento DB2GSE.ST_EXPORT_SHAPE.

Cómo utilizar un geocodificador

El uso de un geocodificador implica la definición del trabajo que desea que realice el geocodificador, configurar éste y ejecutarlo 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. DB2 Spatial Extender puede dar soporte a geocodificadores suministrados por proveedores y suministrados por usuarios.

Es posible que un geocodificador proporcionado por un proveedor convierta las direcciones en coordenadas que DB2 no almacenará, sino que grabará 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 la modalidad de proceso por lotes, un geocodificador intenta, en una sola operación, convertir todos sus datos de entrada desde una sola tabla (o, alternativamente, todas las direcciones de 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. Un geocodificador podría buscar archivos de direcciones conocidas para determinar si cada dirección que recibe como entrada coincide con una dirección conocida en cierto grado. 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.

Tareas del usuario

En DB2 Spatial Extender, as 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 empezar

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

Acerca de esta tarea

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, un geocodificador sólo puede geocodificar los registros que coinciden con sus equivalentes 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 anterior antes de configurar 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 cuadragésimo quinto 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.

Procedimiento

Para configurar operaciones de geocodificación:

Elija el modo en que desea establecer operaciones de geocodificación:

- Emita el mandato db2se setup_gc.
- Ejecute una aplicación que llame al procedimiento DB2GSE.ST_SETUP_GEOCODING.

Qué hacer a continuación

Recomendaciones: Cuando un geocodificador lee un registro de datos de dirección, intenta buscar la coincidencia entre ese registro y un equivalente 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 ortográficas 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 empezar

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.

Acerca de esta tarea

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.

Procedimiento

Para configurar un geocodificador para que se ejecute automáticamente:

Elija qué método desea utilizar para configurar la geocodificación automática:

- Emita el mandato db2se enable_autogc.
- Ejecute una aplicación que llame al procedimiento 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 empezar

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.

Acerca de esta tarea

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.

Procedimiento

Para ejecutar un geocodificador en modalidad de proceso por lotes:

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

- Emita el mandato db2se run_gc.
- Ejecute una aplicación que llame al procedimiento DB2GSE.ST_RUN_GEOCODING.

Capítulo 10. DB2 Spatial Extender en un entorno de base de datos particionada

DB2 Spatial Extender puede utilizarse efectivamente en un entorno de base de datos particionada para realizar el análisis espacial para las tablas de datos de cliente grandes. Los entornos de bases de datos particionadas son útiles para ejecutar las aplicaciones de depósito de datos que puede crear con IBM InfoSphere Warehouse.

Los entornos de bases de datos particionadas dan soporte al proceso de bases de datos de alto rendimiento y alta escalabilidad al permitir el proceso en paralelo en un conjunto de nodos. Cada nodo cuenta con un procesador, memoria y almacenamiento en disco propios.

En un entorno de base de datos particionada, hay tres alternativas disponibles para almacenar tablas de DB2 que contienen columnas espaciales. Puede colocarlas en un solo nodo, distribuirlas entre varios nodos o replicarlas en múltiples nodos. Se da soporte a los índices espaciales para cada alternativa. La opción que seleccione dependerá del tamaño de las tablas y de cómo se utilizarán en las consultas espaciales. Para obtener más información sobre cómo particionar una base de datos, consulte la sección sobre "entornos de bases de datos particionadas" en la publicación *Database Administration Concepts and Configuration Reference*.

Creación y carga de datos espaciales en un entorno de base de datos particionada

La tabla particionada debe tener definida una clave de particionamiento, que se utiliza para controlar la distribución de las filas entre las particiones.

Antes de empezar

Asegúrese de que la base de datos está habilitada para el proceso espacial. Para ello, cree tablas de catálogos espaciales y las funciones y los tipos espaciales en la base de datos. Consulte "Habilitación de una base de datos para operaciones espaciales" en la página 30 para obtener más información.

Acerca de esta tarea

Para optimizar el rendimiento, compruebe que las tablas que contienen columnas espaciales estén distribuidas uniformemente entre las diferentes particiones. Por omisión, la correlación de partición y el algoritmo hash (generación aleatoria) permiten realizar esta acción.

Procedimiento

Para obtener resultados mejores, especifique una o varias columnas en la sentencia CREATE TABLE como clave de particionamiento. Si no especifica una clave de particionamiento, DB2 selecciona la primera columna numérica o de caracteres como clave de particionamiento por omisión. Este valor por omisión podría no distribuir las filas de forma uniforme entre las diferentes particiones.

© Copyright IBM Corp. 1998, 2012

Ejemplo

En el ejemplo siguiente se muestra cómo crear una tabla para que contenga datos espaciales y cómo utilizar el programa de utilidad de importación DB2 Spatial Extender para especificar la clave de particionamiento:

```
CREATE TABLE myschema.counties (id INTEGER PRIMARY KEY,
   name VARCHAR(20),
   geom db2gse.st_polygon)
   IN nodestbs
   DISTRIBUTE BY HASH(id);

db2se import_shape mydb
-tableName counties
-tableSchema myschema
-spatialColumn shape
-fileName /shapefiles/counties.shp
-messagesFile counties.msg
-createTable 0
-client 1
-srsName NAD83_SRS_1
-commitScope 10000
-idcolumn id
-idColumnIsIdentity 1
```

Mejora del rendimiento de las consultas en los datos espaciales de un entorno particionado

Para maximizar el rendimiento de las consultas en un entorno de base de datos particionada, es necesario ubicar juntas las filas de tabla necesarias para cumplir la condición de unión. Es decir, estas uniones deben utilizar las filas de tabla que existen en un nodo sin tener que acceder a las filas de una tabla, o a partes de una tabla, de otro nodo.

Acerca de esta tarea

Las uniones espaciales se utilizan con frecuencia en aplicaciones que encuentran clientes en polígonos concretos o que determinan el riesgo de inundación de los clientes. A continuación se ofrece un ejemplo de una consulta para determinar el riesgo de inundación:

```
Select
  c.name,
  c.address,
  f.risk
from customers as c,
         floodpoly as f
where db2gse.st_within(c.location, f.polygeom) = 1
```

En este ejemplo, la tabla `customers` es bastante grande y está distribuida entre varias particiones. La información del polígono de inundaciones es pequeña. Para implementar la consulta de forma eficaz, asegúrese de que está disponible toda la tabla de polígono de inundaciones en cada partición que contenga datos del cliente.

Procedimiento

1. Importe los datos del polígono a una única partición.

2. Cree una tabla de consulta materializada (MQT) que se replique en las particiones que contienen datos de cliente. Por ejemplo, una vez creada y cargada la tabla `floodpoly`, puede crear la MQT replicada con una sentencia como la siguiente:

```
CREATE TABLE floodpoly
AS ( SELECT * FROM  floodpoly)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
ENABLE QUERY OPTIMIZATION
MAINTAINED BY SYSTEM
DISTRIBUTE BY REPLICATION IN nodestbs

REFRESH TABLE floodpoly;
```

   La MQT replicada con una columna espacial puede tener mantenimiento del usuario o del sistema, pero debe utilizar la opción REFRESH DEFERRED, no REFRESH IMMEDIATE.

3. Para que el compilador de consultas de DB2 seleccione la MQT en lugar de la tabla base, puede establecer los parámetros de DB2 siguientes:

```
UPDATE DB CFG USING DFT_REFRESH_AGE ANY;
UPDATE DB CFG USING DFT_MTTB_TYPES ALL;
```

   Para obtener más información sobre estos parámetros, consulte el "mandato UPDATE DATABASE CONFIGURATION" en la publicación *Consulta de mandatos*.

Qué hacer a continuación

Utilice las herramientas de Explain de DB2 para determinar la eficiencia con la que se ejecutará la consulta.

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

Utilice índices y vistas para consultar las columnas espaciales.

Antes de consultar columnas espaciales, debe tener información detallada sobre la creación de índices y vistas para acceder a dichas columnas. Para crear este tipo de índices, debe conocer la naturaleza de los índices que Spatial Extender utiliza para activar el acceso a los datos espaciales.

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 incorporado. 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.

© Copyright IBM Corp. 1998, 2012

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 uso de funciones espaciales en la cláusula WHERE hace que el optimizador de DB2 tenga en cuenta los índices reticulares espaciales para el plan de acceso.

Las funciones espaciales que tienen este efecto sobre el optimizador de DB2 son:

- 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 siguientes).
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 lenguaje 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 reticulares

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 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.

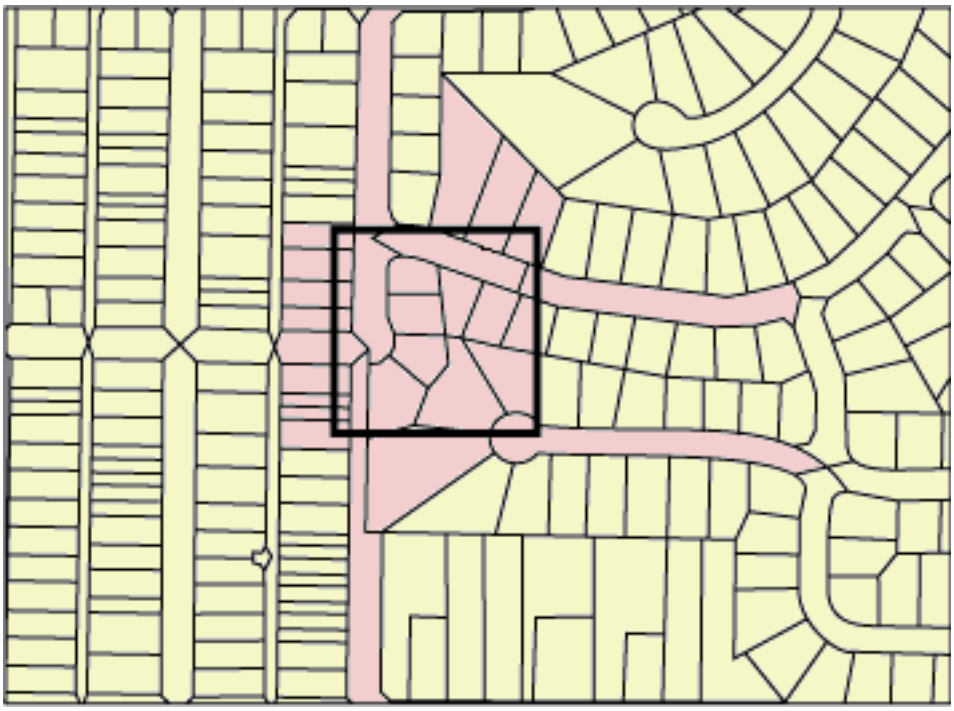


Figura 13. Parcelas de terreno en un vecindario

La Figura 14 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 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.

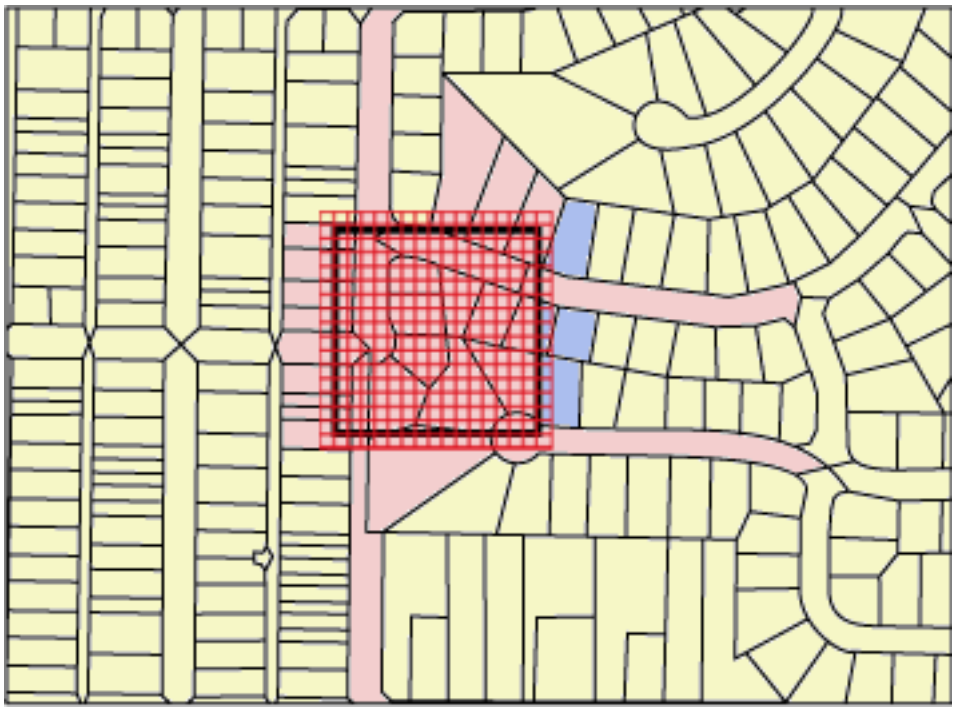


Figura 14. Tamaño de retícula pequeño (25) en parcelas de terreno

Figura 15 en la página 80 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.

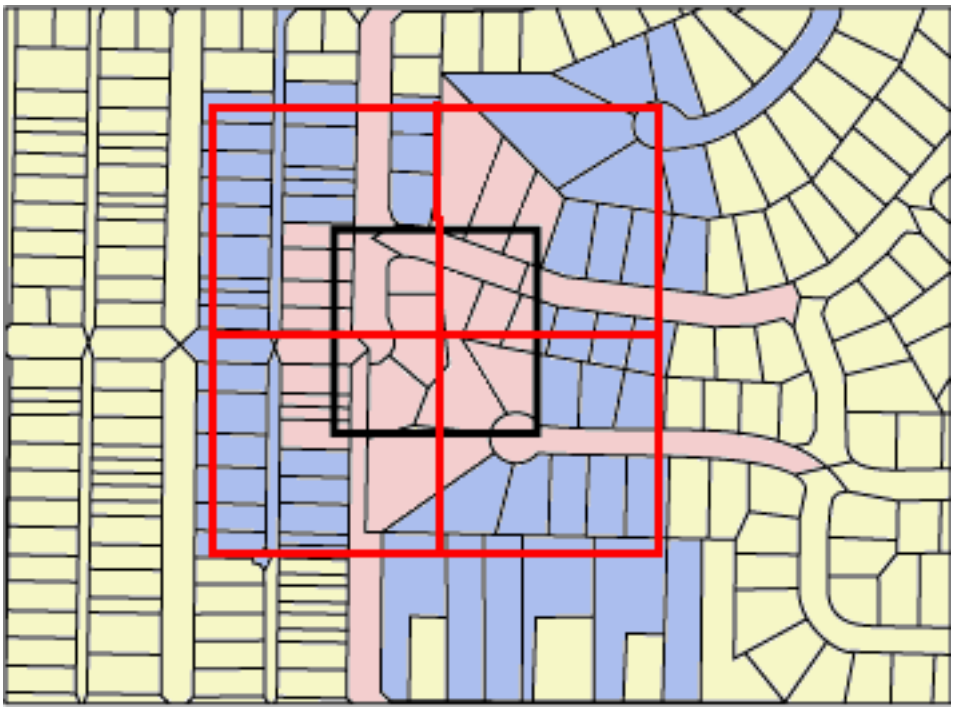


Figura 15. Tamaño de retícula grande (400) en parcelas de terreno

La Figura 16 en la página 81 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.

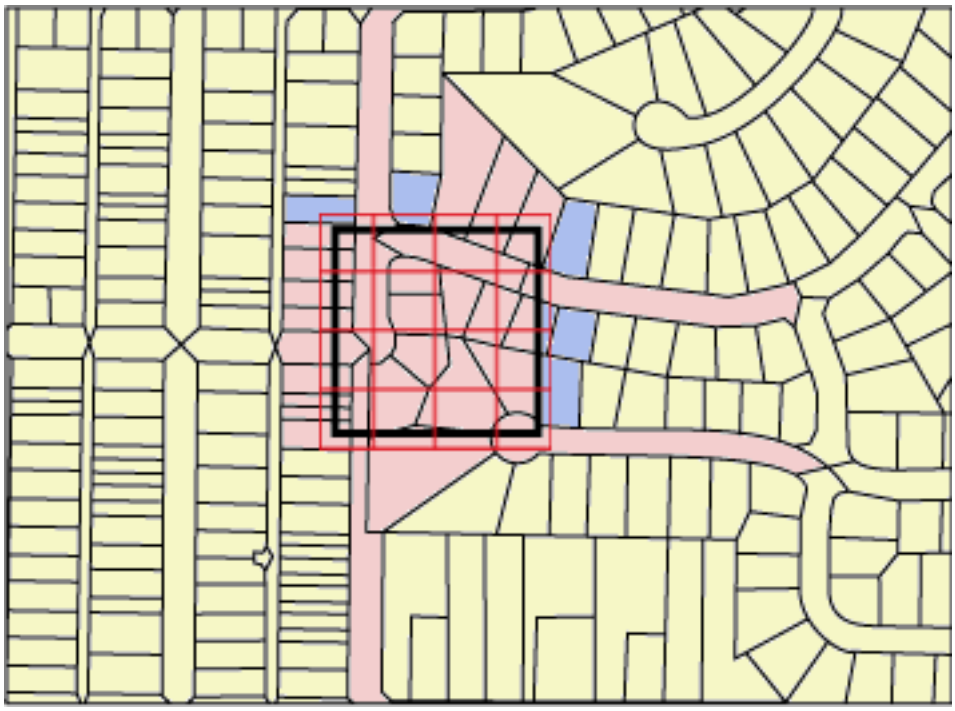


Figura 16. Tamaño de retícula medio (100) en parcelas de terreno

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.

Antes de empezar

Para crear un índice reticular espacial:

- El ID de usuario debe disponer de las autorizaciones necesarias para la sentencia CREATE INDEX de SQL.
- 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.

Acerca de esta tarea

Puede crear índices reticulares espaciales para mejorar el rendimiento de las consultas en las columnas espaciales. Cuando crea un índice reticular espacial, 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.

En esta tarea se describen los pasos para crear índices reticulares espaciales utilizando la sentencia CREATE INDEX de SQL. También puede crear índices reticulares espaciales utilizando una herramienta GIS que funciona con DB2 Spatial Extender. Para obtener información sobre la utilización de una herramienta GIS para crear un índice reticular espacial, consulte la documentación proporcionada con dicha herramienta.

Procedimiento

Para crear índices reticulares espaciales:

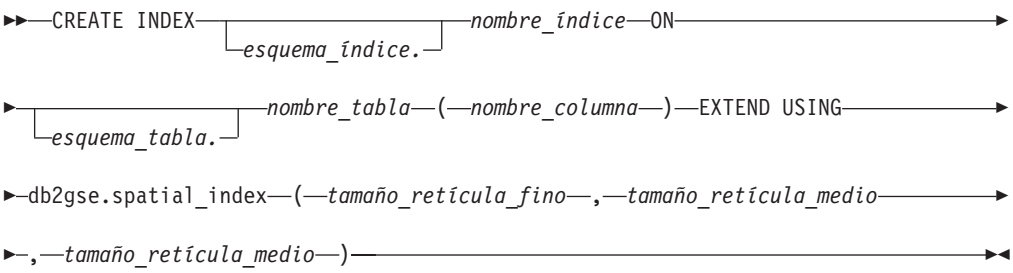
Emita el mandato CREATE INDEX en el procesador de línea de mandatos de DB2 especificando la extensión de índice reticular db2gse.spatial_index con la cláusula EXTEND USING. En el ejemplo siguiente se muestra la creación del índice reticular espacial TERRIDX para la tabla BRANCHES que tiene una columna espacial TERRITORY.

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

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



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 de retícula espacial utilizando la sentencia CREATE INDEX, la validez de los tamaños de retícula se comprueba al indexarse 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

DB2 Spatial Extender proporciona el programa de utilidad Index Advisor como ayuda para optimizar el acceso a los datos espaciales.

Utilice Index Advisor 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

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

Antes de empezar

- 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.

Procedimiento

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 este 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 ven-  Tamaños de retícula
      tana de consulta   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

El análisis de las estadísticas sobre un índice reticular espacial existente puede indicarle si el índice es eficiente o si debe sustituirse por un índice más eficiente.

Antes de empezar

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;
```

Acerca de esta tarea

Utilice Index Advisor para obtener estadísticas sobre índices reticulares espaciales existentes. Analice estas estadísticas y determine si debe sustituir algún 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 una herramienta de línea de mandatos como **db2exfmt** en 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.

Procedimiento

Para analizar estadísticas sobre índices reticulares espaciales existentes y determinar si deben sustituirse por índices más eficientes:

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 ven-  Tamaños de retícula
tana de consulta   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.

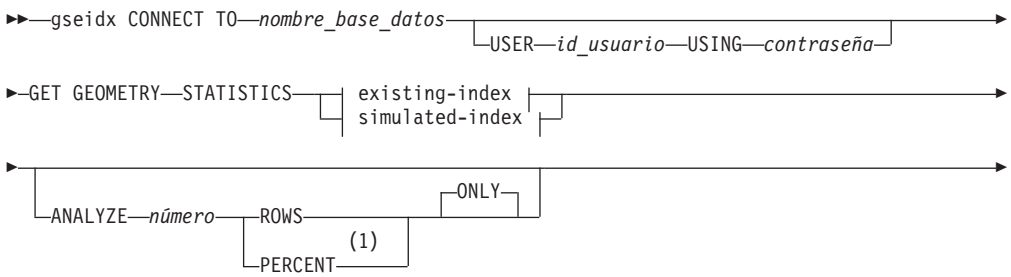
5. Descarte el índice existente y sustitúyalo por un índice que especifique los tamaños de retícula aconsejados. Para el ejemplo del paso anterior, ejecute las siguientes sentencias de DDL:

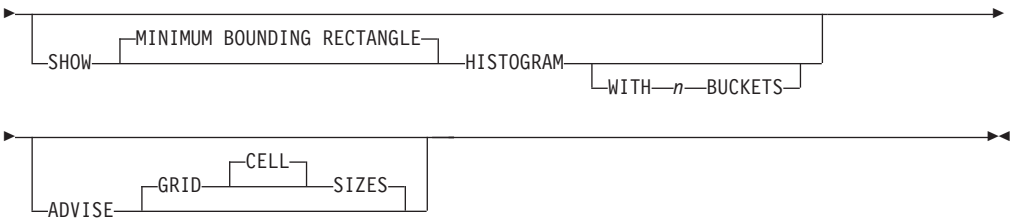
```
DROP INDEX ID_usuario.counties_shape_idx;
CREATE INDEX counties_shape_idx ON ID_usuario.counties(shape) EXTEND USING
  DB2GSE.SPATIAL_INDEX(1.4,3.5,14.0);
```

Mandato gseidx

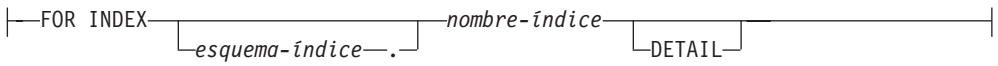
Utilice el mandato **gseidx** para invocar a Index Advisor para los índices reticulares espaciales.

Sintaxis

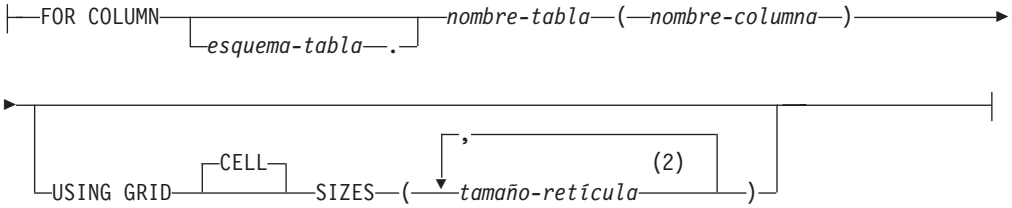




existing-index:



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
Especifique el número o porcentaje aproximado de filas que han de utilizarse para obtener estadísticas para los datos. Si la tabla tiene más de un millón de filas, utilice la cláusula ANALYZE para recopilar estadísticas para un subconjunto de datos de modo que pueda disponer de un tiempo de proceso razonable.

SHOW MINIMUM BOUNDING RECTANGLE HISTOGRAM
Visualiza un diagrama que muestras 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.

Acerca de esta tarea

Esto es especialmente útil cuando una tabla tiene varias columnas espaciales, pues muchas aplicaciones de visualización sólo funcionan con una tabla o vista que tiene una única columna espacial. La definición de vista puede seleccionar la columna espacial adecuada y otras columnas no espaciales que han de estar disponibles para la aplicación.

Capítulo 12. Análisis y generación de información espacial

El análisis y la generación de información espacial requiere que se conozcan los entornos en los que puede emitir consultas y las directrices de uso de las funciones espaciales junto con los índices espaciales. Consulte los ejemplos de los diversos tipos de funciones espaciales que puede invocar en una consulta para obtener información sobre las prestaciones que Spatial Extender ofrece.

Entornos para realizar análisis espaciales

Puede realizar análisis espaciales utilizando SQL y funciones espaciales en el procesador de línea de mandatos de DB2 y en programas de aplicación escritos en cualquier lenguaje soportado por DB2.

Ejemplos del funcionamiento de las funciones espaciales

DB2 Spatial Extender proporciona funciones que realizan diversas operaciones sobre datos espaciales. Estas funciones se pueden clasificar según el tipo de operación que realizan.

La Tabla 2 indica estas categorías, junto con los ejemplos. El texto que sigue a la Tabla 2 muestra la codificación correspondiente a los ejemplos.

Tabla 2. Funciones espaciales y operaciones

Categoría de función	Ejemplo de operación
Devuelve información sobre geometrías determinadas.	Devolver la extensión, en millas cuadradas, del área de ventas de la Tienda 10.
Realiza comparaciones.	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.	Obtener el área de ventas de una tienda a partir de su ubicación.
Convierte geometrías entre formatos de intercambio de datos.	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 saleas_area..ST_Area('STATUTE MILE')
FROM   stores
WHERE id = 10
```

© Copyright IBM Corp. 1998, 2012

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 los índices reticulares 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 3 en la página 95 muestra qué funciones de comparación pueden utilizar un índice reticular espacial:

Tabla 3. Funciones de comparación que pueden utilizar un índice reticular espacial

Función de comparación	Puede utilizar un índice reticular espacial
EnvelopesIntersect	Sí
ST_Contains	Sí
ST_Crosses	Sí
ST_Distance	Sí
ST_EnvIntersects	Sí
ST_Equals	Sí
ST_Intersects	Sí
ST_MBRIntersects	Sí
ST_Overlaps	Sí
ST_Touches	Sí
ST_Within	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 4 se muestran las formas correctas e incorrectas de crear consultas espaciales para utilizar un índice espacial.

Tabla 4. 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

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) = 1`
	En este ejemplo no se infringe ninguna condición.

Tabla 4. Demostración de cómo las funciones espaciales se pueden cumplir y violar normas para utilizar un índice espacial. (continuación)

Consultas que hacen uso de funciones espaciales	Normas vulneradas

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Length(s.location) > 10`
	La función espacial ST_Length no compara geometrías y no puede utilizar un índice espacial.

`SELECT *`
`FROM stores AS s`
`WHERE 1=db2gse.ST_Within(s.location,:BayArea)`
	La función debe ser una expresión situada en el lado izquierdo del predicado.

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) <> 0`
	Las comparaciones de igualdad deben utilizar la constante entera 1.

`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`
	No existe ningún índice espacial en ninguno de los argumentos de la función, por lo que no se puede utilizar ningún índice.

Capítulo 13. Escritura de aplicaciones y utilización del programa de ejemplo

Para grabar aplicaciones para Spatial Extender, debe comprender los requisitos y revisar el programa de ejemplo.

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.

Acerca de esta tarea

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.

Procedimiento

Para incluir el archivo de cabecera de DB2 Spatial Extender en aplicaciones espaciales:

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 en el ejemplo siguiente:

   ```
   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.

© Copyright IBM Corp. 1998, 2012

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 llamar a los procedimientos almacenados de DB2 Spatial Extender desde una aplicación:

1. Llame al procedimiento almacenado DB2GSE.ST_ENABLE_DB para habilitar una base de datos para las 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 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 *nombre_procedimiento_almacenado* es el nombre del procedimiento almacenado. No incluya un signo de exclamación en la llamada.

   ```
   CALL DB2GSE.nombre_procedimiento_almacenado
   ```

   Los procedimientos almacenados de DB2 Spatial Extender en muestran en la tabla siguiente.

Tabla 5. Procedimientos almacenados de DB2 Spatial Extender

Procedimiento almacenado	Descripción
ST_ALTER_COORDSYS	Actualiza un atributo de un sistema de coordenadas en la base de datos.
ST_ALTER_SRS	Actualiza un atributo de un sistema de referencia espacial en la base de datos.
ST_CREATE_COORDSYS	Crea un sistema de coordenadas en la base de datos.
ST_CREATE_SRS	Crea un sistema de referencia espacial en la base de datos.
ST_DISABLE_AUTOGEOCODING	Especifica que DB2 Spatial Extender dejará de sincronizar una columna geocodificada con sus columnas de geocodificación asociadas.
ST_DISABLE_DB	Elimina recursos que permiten a DB2 Spatial Extender almacenar datos espaciales y dar soporte a operaciones que se realizan sobre estos datos.
ST_DROP_COORDSYS	Suprime un sistema de coordenadas de la base de datos.
ST_DROP_SRS	Suprime un sistema de referencia espacial de la base de datos.
ST_ENABLE_AUTOGEOCODING	Especifica que DB2 Spatial Extender sincronizará una columna geocodificada con sus columnas de geocodificación asociadas.

Tabla 5. Procedimientos almacenados de DB2 Spatial Extender (continuación)

Procedimiento almacenado	Descripción
ST_ENABLE_DB	Proporciona una base de datos con los recursos que necesita para almacenar datos espaciales y dar soporte a operaciones.
ST_EXPORT_SHAPE	Exporta los datos seleccionados en la base de datos a un archivo de formas.
ST_IMPORT_SHAPE	Importa un archivo de formas a una base de datos.
ST_REGISTER_GEOCODER	Registra un geocodificador que no sea DB2SE_USA_GEOCODER, que forma parte del producto DB2 Spatial Extender.
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.
ST_UNREGISTER_SPATIAL_COLUMN	Elimina el registro de una columna espacial.

Programa de ejemplo de DB2 Spatial Extender

El programa de ejemplo de DB2 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.

La ubicación del programa runGseDemo varía en función del sistema operativo en el que se haya instalado DB2 Spatial Extender.

- En UNIX, puede localizar el programa runGseDemo en la siguiente vía de acceso:

  ```
  $HOME/sqllib/samples/extenders/spatial
  ```

  donde `$HOME` es el directorio inicial del propietario de la instancia.
- En Windows®, puede localizar el programa runGseDemo en la vía de acceso siguiente:

  ```
  c:\Archivos de programa\IBM\sqllib\samples\extenders\spatial
  ```

  donde `c:\Archivos de programa\IBM\sqllib` 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 6. Pasos del programa de ejemplo de DB2 Spatial Extender

Pasos	Acción y descripción
Habilitar o inhabilitar la base de datos espacial	• 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. • 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. • Habilitar la base de datos espacial Igual que en la acción anterior.

Tabla 6. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Crear o descartar un sistema de coordenadas	• Crear un sistema de coordenadas llamado NORTH_AMERICAN Este paso crea un nuevo sistema de coordenadas en la base de datos. • Descartar el sistema de coordenadas llamado NORTH_AMERICAN Este paso descarta el sistema de coordenadas NORTH_AMERICAN de la base de datos. • 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.
Crear o descartar un sistema de referencia espacial	• 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. • 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. • Crear el SRS llamado KY_STATE_SRS
Crear y llenar las tablas espaciales	• Crear la tabla CUSTOMERS • Llenar la tabla CUSTOMERS La tabla CUSTOMERS representa los datos comerciales que se han almacenado en la base de datos durante varios años. • 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. • 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.

Tabla 6. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Llenar las columnas	• 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. • 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. • 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. • Crear y cargar la tabla REGIONS desde el archivo de formas utilizando el sistema de referencia espacial KY_STATE_SRS
Registrar o desregistrar el geocodificador	• Registrar el geocodificador llamado SAMPLEGC • Desregistrar el geocodificador llamado SAMPLEGC • Registrar el geocodificador KY_STATE_GC 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.
Crear índices espaciales	• Crear el índice reticular espacial para la columna LOCATION de la tabla CUSTOMERS • Descartar el índice reticular espacial para la columna LOCATION de la tabla CUSTOMERS • Crear el índice reticular espacial para la columna LOCATION de la tabla CUSTOMERS • Crear el índice reticular espacial para la columna LOCATION de la tabla OFFICES • Crear el índice reticular espacial para la columna LOCATION de la tabla FLOODZONES • Crear el índice reticular espacial para la columna LOCATION de la tabla REGIONS Estos pasos crean el índice reticular espacial para las tablas CUSTOMERS, OFFICES, FLOODZONES y REGIONS.

Tabla 6. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Habilitar la geocodificación automática.	• 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. • 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, LATITUDE y LONGITUDE de la tabla CUSTOMERS se sincronicen entre sí para operaciones subsiguientes de inserción y actualización.
Realizar operaciones de inserción, actualización y supresión sobre la tabla CUSTOMERS	Estos pasos muestran operaciones de inserción, actualización y supresión sobre las columnas LATITUDE, LONGITUDE, 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. • Insertar algunos registros con una calle diferente • Actualizar algunos registros con una nueva dirección • Suprimir todos los registros de la tabla
Inhabilitar la geocodificación automática	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. • Inhabilitar la geocodificación automática para la columna LOCATION de la tabla CUSTOMERS • Eliminar la definición de geocodificación para la columna LOCATION de la tabla CUSTOMERS • Descartar el índice espacial para la columna LOCATION de la tabla CUSTOMERS **Recomendación:** 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.
Crear una vista y registrar la columna espacial en la vista	Estos pasos crean una vista y registran su columna espacial. • Crear una vista denominada HIGHRISKCUSTOMERS basada en la unión de las tablas CUSTOMERS y FLOODZONES • Registrar la columna espacial de la vista

Tabla 6. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Realizar el análisis espacial	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. • Encontrar el número de clientes atendidos en cada región (ST_Within) • 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) • Para cada región, encontrar los ingresos medios y la prima de cada cliente (ST_Within) • Encontrar el número de zonas con riesgo de inundación que están dentro de cada zona de oficinas (ST_Overlaps) • 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) • Encontrar los clientes cuya ubicación está cerca del límite de una zona determinada con riesgo de inundación (ST_Buffer, ST_Intersects) • Encontrar los clientes expuestos a un riesgo alto que están dentro de una distancia especificada respecto a una oficina determinada (ST_Within) Todos estos pasos hacen uso de la función interna gseRunSpatialQueries.
Exportar datos espaciales a archivos de forma	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). • Exportar la vista HIGHRISKCUSTOMERS a archivos de forma

Capítulo 14. Identificación de problemas de DB2 Spatial Extender

Para identificar un problema de DB2 Spatial Extender, debe 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

Saber cómo se estructuran los mensajes de DB2 Spatial Extender y cómo obtener información adicional sobre estos mensajes pueden ayudarle a determinar si la operación espacial solicitada se ha ejecutado satisfactoriamente o si ha producido un error.

Puede trabajar con DB2 Spatial Extender a través de cualquiera de las interfaces siguientes:

- Procedimientos almacenados de DB2 Spatial Extender
- Funciones de DB2 Spatial Extender
- Procesador de línea de mandatos (Command Line Processor, CLP) de DB2 Spatial Extender

Todas estas interfaces devuelven mensajes de DB2 Spatial Extender.

La tabla siguiente describe cada parte del texto de este mensaje de ejemplo de DB2 Spatial Extender:

```
GSE0000I:  La operación se ejecutó satisfactoriamente.
```

Tabla 7. 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.

© Copyright IBM Corp. 1998, 2012

Tabla 7. 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 "? GSEnnnn" | 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

Utilice los parámetros de salida de los procedimientos almacenados de DB2 Spatial Extender para diagnosticar problemas cuando se invocan explícitamente procedimientos almacenados en programas de aplicación o desde la línea de mandatos de DB2.

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 8. 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 consta del 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".

Para diagnosticar los procedimientos almacenados a los que se ha llamado implícitamente mediante mandatos de DB2 Spatial Extender, utilice los mensajes que devuelve el CLP de DB2 Spatial Extender. Para obtener más detalles, consulte: “Cómo interpretar mensajes de DB2 Spatial Extender” en la página 105

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.
- Emprender 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 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 "<número-fila>".
            El identificador numérico del sistema de referencia espacial
            es "<id-srs>".

Explicación: En la fila número-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 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.

Los mensajes siguientes son ejemplos que indican la existencia de un error y de un aviso:

- 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 9. 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 9. Partes significativas de los mensajes de las funciones de DB2 Spatial Extender (continuación)

Parte del mensaje	Descripción
SQLSTATE=38SSL	Código SQLSTATE que identifica con más detalle el error. Se emite un código SQLSTATE para cada sentencia o fila. • Los códigos SQLSTATE para errores de funciones de Spatial Extender tienen la forma 38S*xx*, donde *x* es una letra o un número. • Los códigos SQLSTATE para avisos de funciones de Spatial Extender tienen la forma 01HS*x*, donde *x* es una letra o un número.

Ejemplo de mensaje de error SQL0443

Suponga que intenta insertar los valores de un polígono en la tabla POLYGON_TABLE, tal como se muestra en la sentencia siguiente:

```
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 este ejemplo, 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

Los mensajes del CLP de DB2 Spatial Extender indican si una operación se ha ejecutado correcta o incorrectamente. Asimismo, pueden proporcionar información sobre formas.

El CLP de DB2 Spatial Extender 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 actualización.
- Operaciones de importación y exportación de formas con el cliente.

Ejemplos de mensajes para procedimientos almacenados que el CLP de DB2 Spatial Extender devuelve

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 consta del 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 para información sobre formas que el CLP de Spatial Extender devuelve

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",0.0174532925199433295]]"
```

Ejemplos de mensajes para operaciones de actualización que el CLP de Spatial Extender devuelve

Cuando invoca mandatos que realizan operaciones de actualización, se devuelven mensajes que indican si la operación se ha ejecutado correcta o incorrectamente.

Supongamos que actualiza la base de datos habilitada para operaciones espaciales *mydb* mediante 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.
```

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.

Antes de empezar

Asegúrese de que dispone de la autorización adecuada para emitir el mandato **db2trc**. En los sistemas operativos UNIX, debe tener autorización SYSADM, SYSCTRL o SYSMAINT para rastrear una instancia de DB2. En los sistemas operativos Windows no es necesaria ninguna autorización especial.

Acerca de esta tarea

El recurso de rastreo DB2 se activa mediante el mandato **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.

Restricciones

- Active este recurso solo cuando se lo indique el servicio técnico de DB2.
- 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.

Procedimiento

Para solucionar problemas en DB2 Spatial Extender utilizando el recurso de rastreo de DB2:

1. Cierre todas las demás aplicaciones.
2. Active el rastreo.

   El servicio técnico de DB2 le proporciona los parámetros específicos para este paso. El mandato básico es:

   ```
   db2trc on
   ```

   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. Vuelque 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 manera:
   - 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 el sistema de base de datos DB2 así como por 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 operativos Windows, 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 el gestor de bases de datos DB2 registra en el archivo de 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 15. Vistas de catálogo

Puede utilizar vistas de catálogo para que Spatial Extender obtenga información útil acerca de sus datos espaciales.

Las vistas de catálogo de Spatial Extender contienen información acerca de:

"Vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS"
Sistemas de coordenadas que puede utilizar

"Vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS" en la página 116
Columnas espaciales que puede llenar o actualizar

"Vista de catálogo DB2GSE.ST_GEOCODERS" en la página 119 y "Vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS" en la página 117
Geocodificadores que puede utilizar

"Vista de catálogo DB2GSE.ST_GEOCODING" en la página 119 y "Vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS" en la página 121
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 122
Longitudes máximas permitidas que puede asignar a las variables.

"Vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" en la página 123 Sistemas de referencia espacial que puede utilizar.

"Vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE" en la página 126
Las unidades de medida (metros, millas, pies, etc.) en las que se pueden generar las funciones espaciales.

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 10. Columnas de la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS

Nombre	Tipo de datos	Posibilidad de nulos	Contenido
COORDSYS_NAME	VARCHAR(128)	No	Nombre de este sistema de coordenadas. El nombre es exclusivo dentro de la base de datos.

© Copyright IBM Corp. 1998, 2012

Tabla 10. Columnas de la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS (continuación)

Nombre	Tipo de datos	Posibilidad de nulos	Contenido
COORDSYS_TYPE	VARCHAR(128)	No	Tipo de este sistema de coordenadas: **PROYECTADO** Bidimensional. **GEOGRÁFICO** Tridimensional. Utiliza coordenadas X e Y. **GEOCÉNTRICO** Tridimensional. Utiliza coordenadas X, Y y Z. **SIN ESPECIFICAR** Sistema de coordenadas abstracto o que no es del mundo real. El valor de esta columna se obtiene de la columna DEFINITION.
DEFINITION	VARCHAR(2048)	No	Representación de WKT (texto convencional) de la definición de este sistema de coordenadas.
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.

Si se ha registrado una columna espacial en asociación con un sistema de referencia espacial y se ha especificado **compute_extents**, también puede utilizar la vista para consultar información sobre las extensiones geográficas correspondientes a la columna espacial y la fecha en la que se calcularon por última vez.

Para ver una descripción de DB2GSE.ST_GEOMETRY_COLUMNS, consulte la tabla siguiente.

Tabla 11. Columnas de la vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS

Nombre	Tipo de datos	Posibilidad de nulos	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.
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.
MIN_X	DOUBLE	Sí	Valor *x* mínimo de todos los valores espaciales de la columna.
MIN_Y	DOUBLE	Sí	Valor *y* mínimo de todos los valores espaciales de la columna.
MIN_Z	DOUBLE	Sí	Valor *z* mínimo de todos los valores espaciales de la columna.
MAX_X	DOUBLE	Sí	Valor *x* máximo de todos los valores espaciales de la columna.
MAX_Y	DOUBLE	Sí	Valor *y* máximo de todos los valores espaciales de la columna.
MAX_Z	DOUBLE	Sí	Valor *z* máximo de todos los valores espaciales de la columna.
MIN_M	DOUBLE	Sí	Valor *m* mínimo de todos los valores espaciales de la columna.
MAX_M	DOUBLE	Sí	Valor *m* máximo de todos los valores espaciales de la columna.
EXTENT_TIME	TIMESTAMP	Sí	Indicación de la fecha y la hora en que se calcularon las extensiones por última vez.

Vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS

Utilice la vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS para recuperar información sobre los parámetros de los geocodificadores registrados.

Para obtener más información sobre los parámetros de los geocodificadores, consulte la vista de catálogo SYSCAT.ROUTINEPARMS.

Para ver una descripción de las columnas de esta vista, consulte la tabla siguiente.

Tabla 12. Columnas de DB2GSE.ST_GEOCODER_PARAMETERS

Nombre	Tipo de datos	Posibilidad de nulos	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 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.
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 parámetros de entrada del procedimiento 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 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 obtener información sobre los parámetros de los geocodificadores, consulte la vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS y la vista de catálogo SYSCAT.ROUTINEPARMS. Para obtener información sobre las funciones que se utilizan como geocodificadores, consulte la vista de catálogo SYSCAT.ROUTINES.

Para ver una descripción de las columnas de la vista DB2GSE.ST_GEOCODERS, consulte la tabla siguiente.

Tabla 13. Columnas de la vista de catálogo DB2GSE.ST_GEOCODERS

Nombre	Tipo de datos	Posibilidad de nulos	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.
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 de este parámetro de salida de geocodificador. Este nombre se obtiene del catálogo de DB2.
RETURN_TYPE_NAME	VARCHAR(128)	No	Nombre no calificado del tipo de datos de este parámetro de salida de 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 14 en la página 120, 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, 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 14. Columnas de la vista de catálogo DB2GSE.ST_GEOCODING

Nombre	Tipo de datos	Posibilidad de nulos	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.
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: **PROCESO POR LOTES** Sólo está habilitada la geocodificación de proceso por lotes. **AUTOMÁTICA** La geocodificación automática está configurada y activada. **NO VÁLIDA** 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 están disponibles a través de la vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS.

Por ejemplo, una operación puede ser la comparación de direcciones proporcionadas como entrada para los datos de referencia.

Cuando configura operaciones para un geocodificador, especifica cuál será el 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 15. Columnas de la vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS

Nombre	Tipo de datos	Posibilidad de nulos	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. Los valores combinados en las columnas TABLE_SCHEMA, TABLE_NAME y COLUMN_NAME identifican de forma exclusiva esta columna espacial.
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 geocodificador para la columna identificada en la columna COLUMN_NAME. Un registro en la vista de catálogo SYSCAT.ROUTINEPARMS 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.

Tabla 15. Columnas de la vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS (continuación)

Nombre	Tipo de datos	Posibilidad de nulos	Contenido
PARAMETER_NAME	VARCHAR(128)	Sí	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. Este contenido de la columna PARAMETER_NAME se obtiene del catálogo de DB2.
PARAMETER_VALUE	VARCHAR(2048)	Sí	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. 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.

Vista de catálogo DB2GSE.ST_SIZINGS

Utilice la vista de catálogo DB2GSE.ST_SIZINGS para recuperar información sobre las variables soportadas y su longitud máxima.

La vista de catálogo devuelve la información siguiente:

- 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 de esta vista, consulte la tabla siguiente.

Tabla 16. Columnas de la vista de catálogo DB2GSE.ST_SIZINGS

Nombre	Tipo de datos	Posibilidad de nulos	Contenido
VARIABLE_NAME	VARCHAR(128)	No	Término que indica una variable. El término es exclusivo dentro de la base de datos.

Tabla 16. Columnas de la vista de catálogo DB2GSE.ST_SIZINGS (continuación)

Nombre	Tipo de datos	Posibilidad de nulos	Contenido
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: **Un valor numérico distinto 0** La longitud máxima permitida de valores asignados a esta variable. **0** Se permite cualquier longitud o no se puede determinar la longitud permitida. **NULL** 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 las operaciones espaciales, registra cinco sistemas de referencia espacial por omisión.
- 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 17. Columnas de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS

Nombre	Tipo de datos	Posibilidad de nulos	Contenido
SRS_NAME	VARCHAR(128)	No	Nombre del sistema de referencia espacial. Este nombre es exclusivo dentro de la base de datos.

Tabla 17. Columnas de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (continuación)

Nombre	Tipo de datos	Posibilidad de nulos	Contenido
SRS_ID	INTEGER	No	Identificador numérico del sistema de referencia espacial Cada sistema de referencia espacial tiene un identificador numérico exclusivo. Las funciones espaciales especifican sistemas de referencia por sus identificadores numéricos en lugar de hacerlo por los nombres.
X_OFFSET	DOUBLE	No	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.
M_SCALE	DOUBLE	No	Factor de escala por el que se multiplica el número resultante de sustraer el desplazamiento a una medida.

Tabla 17. Columnas de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (continuación)

Nombre	Tipo de datos	Posibilidad de nulos	Contenido
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_ COORDSYS_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_COORDSYS_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

Consulte la vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE para determinar qué unidades de medida puede seleccionar.

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 ver una descripción de las columnas de esta vista, consulte la tabla siguiente.

Tabla 18. Columnas de la vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE

Nombre	Tipo de datos	Posibilidad de nulos	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: **LINEAL** La unidad de medida es lineal. **ANGULAR** 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.

Vista de catálogo DB2GSE.SPATIAL_REF_SYS

Utilice la vista de catálogo DB2GSE.SPATIAL_REF_SYS para obtener información acerca de los sistemas de referencia espacial que están registrados en DB2 Spatial Extender.

Importante: Esta vista de catálogo está en desuso y se ha sustituido por la "Vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" en la página 123.

Para ver una descripción de las columnas de esta vista, consulte la tabla siguiente.

Tabla 19. Columnas en la vista de catálogo DB2GSE.SPATIAL_REF_SYS

Nombre	Tipo de datos	Posibilidad de nulos	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.

Tabla 19. Columnas en la vista de catálogo DB2GSE.SPATIAL_REF_SYS (continuación)

Nombre	Tipo de datos	Posibilidad de nulos	Contenido
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).
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.

Capítulo 16. Mandatos de DB2 Spatial Extender

Utilice estos mandatos para configurar DB2 Spatial Extender y desarrollar proyectos que utilicen datos espaciales.

Invocación de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos

Utilice el procesador de línea de mandatos (CLP) de DB2 Spatial Extender, 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 la sección Autorización correspondiente a cada mandato.

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

Para saber qué mandatos y parámetros de db2se están disponibles:

- Escriba `db2se` o `db2se -h`; a continuación, pulse Intro. Se visualizará una lista de submandatos de db2se.
- Escriba `db2se` y un mandato o bien `db2se` y un mandato seguido del parámetro **-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.

Para emitir un mandato de db2se, escriba `db2se`. A continuación, escriba un mandato, seguido de los parámetros y los valores de parámetro que el mandato db2se 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 de usuario 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 \"mytable\" para hacer referencia a la tabla llamada mytable. Si un valor con

© Copyright IBM Corp. 1998, 2012

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"

Importante: No especifique la lista completa de nombres y valores de parámetros entre comillas dobles.

La mayoría de los mandatos de **db2se** ejecutan el procedimiento almacenado correspondiente en el servidor DB2, con la excepción del mandato **db2se shape_info**. Este mandato se ejecuta en el cliente y puede acceder a la información del sistema de referencia espacial y de coordenadas desde una base de datos habilitada para las operaciones espaciales.

Los mandatos **db2se import_shape** y **db2se export_shape** también pueden ejecutarse en el cliente. Esto resulta de utilidad porque permite el acceso a los archivos locales del cliente.

Mandato db2se alter_cs

El mandato **db2se alter_cs** actualiza la definición de un sistema de coordenadas.

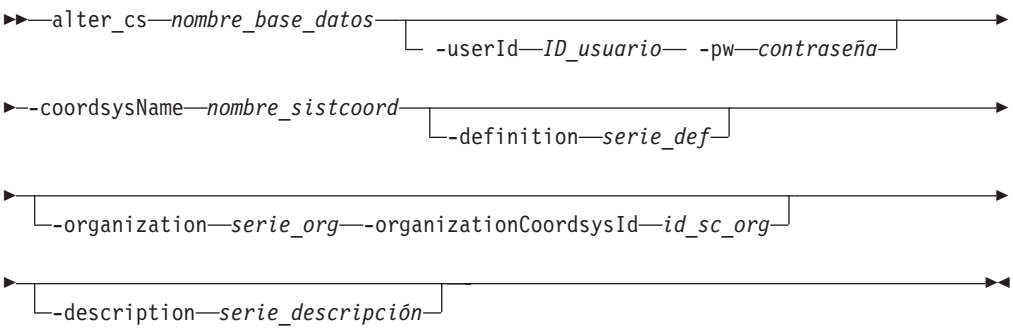
Puede utilizar este mandato para actualizar la serie de definición, el nombre de la organización, el identificador del sistema de coordenadas de la organización y la descripción en un sistema de coordenadas definido mediante el mandato **db2se create_cs** o el procedimiento almacenado ST_CREATE_COORDSYS. La información acerca del sistema de coordenadas está disponible en la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Autorización

El ID de usuario debe tener autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se alter_cs



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea alterar el sistema de coordenadas.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-coordsysName *nombre_sistcoord*
Identifica de forma exclusiva el sistema de coordenadas. La longitud máxima de este parámetro es de 128 caracteres.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

-definition *serie_def*
Define el sistema de coordenadas. El proveedor que suministra el sistema de coordenadas normalmente proporciona la información para este parámetro. La longitud máxima de este parámetro es de 2048 caracteres.

-organization *serie_org*
Define el nombre de la organización que definió el sistema de coordenadas y proporcionó la definición para él; por ejemplo, "European Petroleum Survey Group (EPSG)". La longitud máxima de este parámetro es de 128 caracteres.

La combinación de *serie_org* e *id_sc_org* identifica de forma exclusiva el sistema de coordenadas.

-organizationCoordsysId *id_sc_org*
Especifica un identificador numérico. La entidad que está especificada en *serie_org* asigna este valor. Este valor no es necesariamente exclusivo en todo el sistema de coordenadas.

La combinación de *serie_org* e *id_sc_org* identifica de forma exclusiva el sistema de coordenadas.

-description *serie_descripción*
Describe el sistema de coordenadas explicando su aplicación. La longitud máxima de este parámetro es de 256 caracteres.

Notas sobre el uso

Si utiliza este mandato 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.

Ejemplo

El ejemplo siguiente actualiza la definición de un sistema de coordenadas denominado MYCOORDSYS con un nuevo nombre de organización.

```
db2se alter_cs mydb -coordsysName mycoordsys -organization myNeworganizationb
```

Mandato db2se alter_srs

El mandato **db2se alter_srs** actualiza la definición de un sistema de referencia espacial.

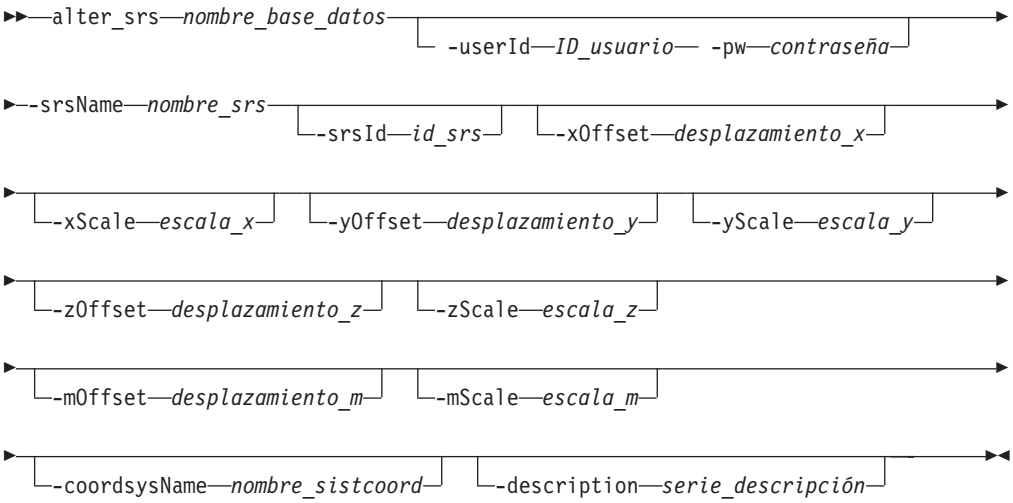
Puede utilizar este mandato para cambiar el offset, la escala o el nombre del sistema de coordenadas de un sistema de referencia espacial. La información acerca del sistema de coordenadas está disponible en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Autorización

El ID de usuario debe tener autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se alter_srs



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea actualizar la definición de un sistema de referencia espacial.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-srsName *nombre_srs*
Identifica el sistema de referencia espacial que desea actualizar. El valor de *nombre_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

-srsId *id_srs*
Identifica de forma exclusiva el sistema de referencia espacial. Este identificador se utiliza como parámetro de entrada para varias funciones espaciales.

-xOffset *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 con representaciones externas (por ejemplo, de texto convencional (WKT), binarias convencionales (WKB) y de forma) a la representación interna de DB2 Spatial Extender.

-xScale *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 que el desplazamiento *desplazamiento_x* se reste cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-yOffset *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 con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-yScale *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 que el desplazamiento *desplazamiento_y* se reste cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender. Este factor de escala debe ser el mismo que *escala_x*.

-zOffset *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 con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-zScale *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 que el desplazamiento *desplazamiento_z* se reste cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-mOffset *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 con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-mScale *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 que el desplazamiento *desplazamiento_m* se reste cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-coordsysName *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 DB2GSE.ST_COORDINATE_SYSTEMS.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles. La longitud máxima de este parámetro es de 128 caracteres.

-description *serie_descripción*
Describe el sistema de coordenadas explicando su aplicación. La longitud máxima de este parámetro es de 256 caracteres.

Notas sobre el uso

Si utiliza este mandato 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.

Ejemplo

El ejemplo siguiente modifica un sistema de referencia espacial denominado misrs una descripción diferente.

```
db2se alter_srs mibd -srsName misrs -description "Esto es mi propio sistema de referencia espacial."
```

El ejemplo siguiente modifica un sistema de referencia espacial denominado MISRS_35 con un xOffset y una descripción diferentes porque no hay datos espaciales que utilicen MISRS_35. Si tiene datos espaciales que lo utilicen, los datos ya no se podrán utilizar.

```
db2se alter_srs mibd -srsName misrs_35 -xOffset 35
        -description "Esto es mi propio sistema de referencia espacial con xOffset=35."
```

Mandato db2se create_cs

El mandato **db2se create_cs** crea un sistema de coordenadas.

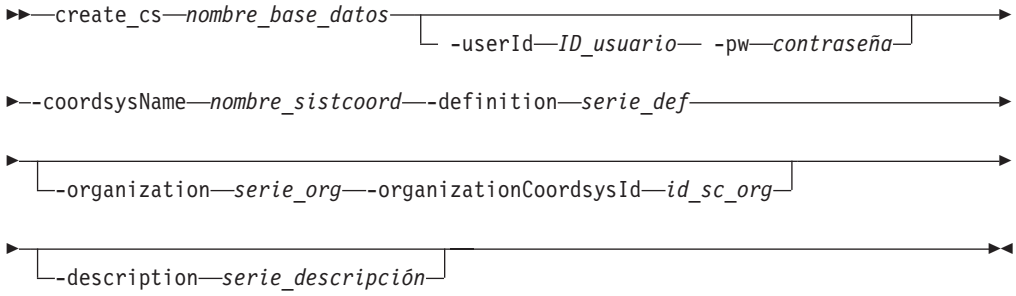
Este mandato almacena información en la base de datos acerca de un sistema de coordenadas nuevo. La información acerca del sistema de coordenadas está disponible en la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Autorización

El ID de usuario debe tener autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se create_cs



Parámetros del mandato

Donde:

nombre_base_datos
Nombre de la base de datos para la que desea crear el sistema de coordenadas.

-userId *ID_usuario*
ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Contraseña del *id_usuario*.

-coordsysName *nombre_sistcoord*
Identifica de forma exclusiva el sistema de coordenadas. La longitud máxima de este parámetro es de 128.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

-definition *serie_def*
Define el sistema de coordenadas. El proveedor que suministra el sistema de coordenadas normalmente proporciona la información para este parámetro. La longitud máxima de este parámetro es de 2048.

-organization *serie_org*
Define el nombre de la organización que definió el sistema de coordenadas y proporcionó la definición para él; por ejemplo, "European Petroleum Survey Group (EPSG)".

La combinación de *serie_org* e *id_sc_org* identifica de forma exclusiva el sistema de coordenadas. La longitud máxima de este parámetro es de 128.

-organizationCoordsysId *id_sc_org*
Especifica un identificador numérico. La entidad que está especificada en *serie_org* asigna este valor. Este valor no es necesariamente exclusivo en todo el sistema de coordenadas.

La combinación de *serie_org* e *id_sc_org* identifica de forma exclusiva el sistema de coordenadas.

-description *serie_descripción*
Describe el sistema de coordenadas explicando su aplicación. La longitud máxima de este parámetro es de 256.

Notas sobre el uso

DB2 Spatial Extender proporciona más de 4.000 sistemas de coordenadas. Rara vez es necesario crear un sistema de coordenadas. Además, necesitará crear un sistema de referencia espacial basado en el nuevo sistema de coordenadas.

Ejemplo

El ejemplo siguiente crea un sistema de coordenadas denominado MYCOORDSYS.

```
db2se create_cs mydb -coordsysName mycoordsys
                     -definition GEOCS[\"GCS_NORTH_AMERICAN_1983\",
                                 DATUM["D_North_American_1983\",
                                 SPHEROID[\"GRS_1980\",6387137,298.257222101]],
                                 PRIMEM[\"Greenwich\",0],
                                 UNIT["Degree\", 0.0174532925199432955]]
```

Mandato db2se create_srs

El mandato **db2se create_srs** crea un sistema de referencia espacial.

Puede utilizar este mandato para crear la definición de 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 el sistema de referencia espacial especificado. Las extensiones son los valores mínimos y máximos de las coordenadas X, Y, Z y M. La información acerca del sistema de coordenadas está disponible en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Este mandato tiene dos métodos para crear sistemas de referencia espacial:

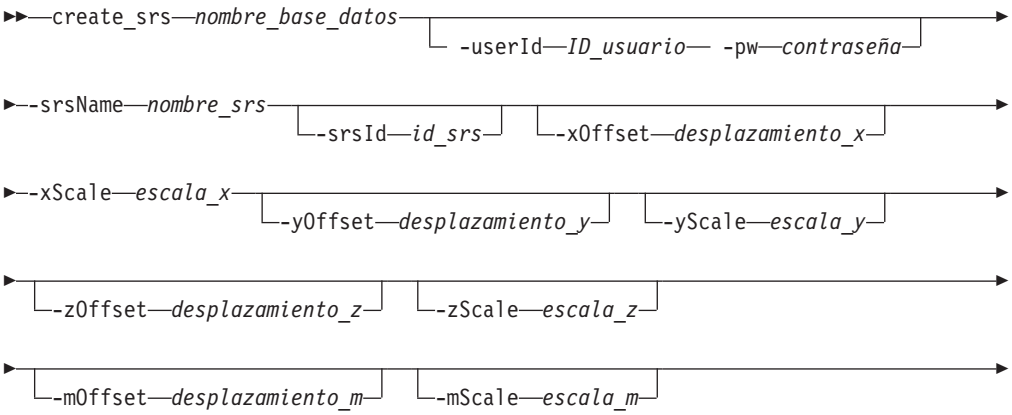
- Utilizar factores de conversión (desplazamientos y factores de escala)
- Utilizar extensiones y precisión (se calculan los factores de conversión)

Autorización

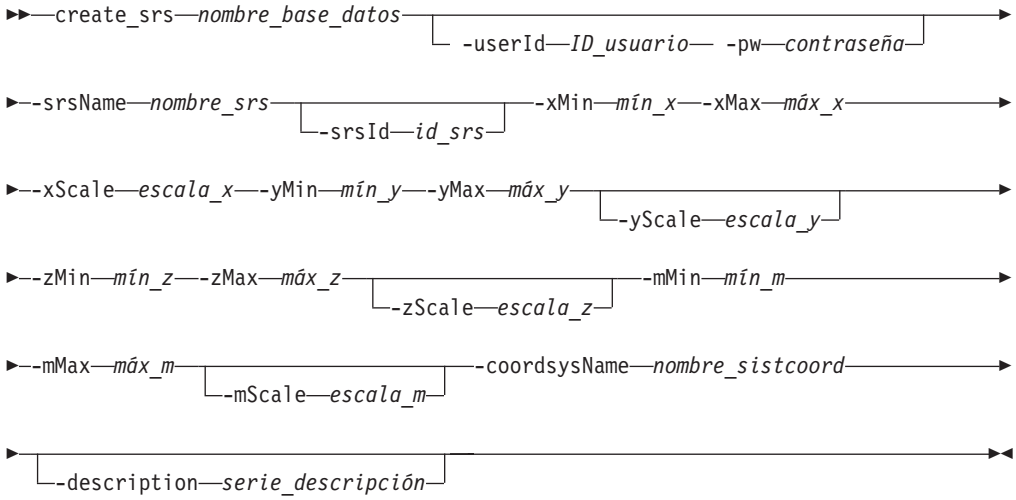
El ID de usuario debe tener autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se create_srs (utilizando factores de conversión)



Mandato db2se create_srs (utilizando extensiones y precisión)



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea crear la definición de un sistema de referencia espacial.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-srsName *nombre_srs*
Identifica el sistema de referencia espacial que desea crear. El valor de *nombre_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

-srsId *id_srs*
Identifica de forma exclusiva el sistema de referencia espacial. Este identificador se utiliza como parámetro de entrada para varias funciones espaciales.

-xMin *mín_x*
Especifica el valor mínimo de la coordenada X para todas las geometrías que utilizan el sistema de referencia espacial especificado.

-xMax *máx_x*
Especifica el valor máximo de la coordenada X para todas las geometrías que utilizan el sistema de referencia espacial especificado. En función del valor de *escala_x*, el valor máximo de la coordenada X devuelto por la vista DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS puede ser mayor que el valor especificado en *máx_x*. El valor devuelto por la vista es correcto.

-xOffset *desplazamiento_x*
Especifica el desplazamiento de todas las coordenadas X de las geometrías que se representan en el sistema de referencia espacial especificado. El desplazamiento se resta antes de que se aplique el factor de escala *escala_x* cuando se convierten las geometrías con representaciones externas (por ejemplo, de texto convencional (WKT), binarias convencionales (WKB) y de forma) a la representación interna de DB2 Spatial Extender.

-xScale *escala_x*
Especifica el factor de escala para todas las coordenadas X de las geometrías que se representan en el sistema de referencia espacial especificado. El factor de escala se aplica (multiplicación) después de que el desplazamiento *desplazamiento_x* se reste cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-yMin *mín_y*
Especifica el valor mínimo de la coordenada Y para todas las geometrías que utilizan el sistema de referencia espacial especificado.

-yMax *máx_y*
Especifica el valor máximo de la coordenada Y para todas las geometrías que utilizan el sistema de referencia espacial especificado. En función del valor de *escala_y*, el valor máximo de la coordenada Y devuelto por la vista DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS puede ser mayor que el valor de *máx_y* especificado. El valor devuelto por la vista es correcto.

-yOffset *desplazamiento_y*
Especifica el desplazamiento de todas las coordenadas Y de las geometrías que se representan en el sistema de referencia espacial especificado. El desplazamiento se resta antes de que se aplique el factor de escala *escala_y* cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-yScale *escala_y*
Especifica el factor de escala para todas las coordenadas Y de las geometrías que se representan en el sistema de referencia espacial especificado. El factor de escala se aplica (multiplicación) después de que el desplazamiento *desplazamiento_y* se reste cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender. Este factor de escala debe ser el mismo que *escala_x*.

-zMin *mín_z*
Especifica el valor mínimo de la coordenada Z para todas las geometrías que utilizan el sistema de referencia espacial especificado.

-zMax *máx_z*
Especifica el valor máximo de la coordenada Z para todas las geometrías que utilizan el sistema de referencia espacial especificado. En función del valor de *escala_z*, el valor máximo de la coordenada Z devuelto por la vista DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS puede ser mayor que el valor de *máx_z* especificado. El valor devuelto por la vista es correcto.

-zOffset *desplazamiento_z*
Especifica el desplazamiento de todas las coordenadas Z de las geometrías que se representan en el sistema de referencia espacial especificado. El desplazamiento se resta antes de que se aplique el factor de escala *escala_z*

cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-zScale *escala_z*
Especifica el factor de escala para todas las coordenadas Z de las geometrías que se representan en el sistema de referencia espacial especificado. El factor de escala se aplica (multiplicación) después de que el desplazamiento *desplazamiento_z* se reste cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-mMin *mín_m*
Especifica el valor mínimo de la coordenada M para todas las geometrías que utilizan el sistema de referencia espacial especificado.

-mMax *máx_m*
Especifica el valor máximo de la coordenada M para todas las geometrías que utilizan el sistema de referencia espacial especificado. En función del valor de *escala_m*, el valor máximo de la coordenada M devuelto por la vista DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS puede ser mayor que el valor de *máx_m* especificado. El valor devuelto por la vista es correcto.

-mOffset *desplazamiento_m*
Especifica el desplazamiento de todas las coordenadas M de las geometrías que se representan en el sistema de referencia espacial especificado. El desplazamiento se resta antes de que se aplique el factor de escala *escala_m* cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-mScale *escala_m*
Especifica el factor de escala para todas las coordenadas M de las geometrías que se representan en el sistema de referencia espacial especificado. El factor de escala se aplica (multiplicación) después de que el desplazamiento *desplazamiento_m* se reste cuando se convierten las geometrías con representaciones externas (por ejemplo, WKT, WKB y de forma) a la representación interna de DB2 Spatial Extender.

-coordsysName *nombre_sistcoord*
Identifica de forma exclusiva el sistema de coordenadas en el que se basa el sistema de referencia espacial especificado. El sistema de coordenadas debe estar listado en la vista DB2GSE.ST_COORDINATE_SYSTEMS.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles. La longitud máxima de este parámetro es de 128.

-description *serie_descripción*
Describe el sistema de coordenadas explicando su aplicación. La longitud máxima de este parámetro es de 256.

Notas sobre el uso

DB2 Spatial Extender incluye soporte para cinco sistemas de referencia espacial y más de 4.000 sistemas de coordenadas para elegir. Rara vez es necesario crear un sistema de referencia espacial.

Ejemplo

El ejemplo siguiente crea un sistema de referencia espacial denominado MYSRS utilizando la sintaxis del mandato para los factores de conversión.

```
db2se create_srs mydb -srsName mysrs -srsID 100
        -xOffset -180 -xScale 1000000 -yOffset -90
        -coordsysName \"GCS_North_American_1983\"
```

Mandato db2se disable_autogc

El mandato **db2se disable_autogc** inhabilita la geocodificación automática.

Este mandato impide que DB2 Spatial Extender sincronice una columna geocodificada con las 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. La información sobre las columnas geocodificadas está disponible en la vista de catálogo DB2GSE.ST_GEOCODING.

Autorización

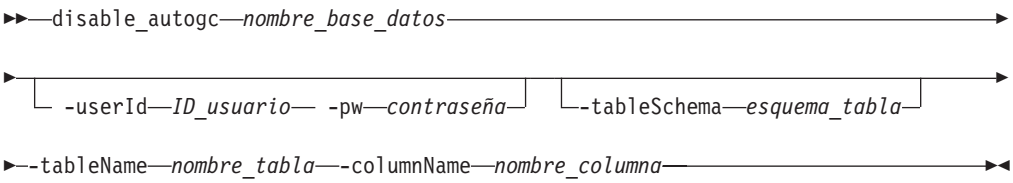
El ID de usuario debe tener uno de los privilegios o autorizaciones siguientes para ejecutar este mandato:

- 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 y autorización DROPIN sobre el esquema DB2GSE.
- Privilegios ALTER y UPDATE sobre esta tabla y autorización DROPIN sobre el esquema DB2GSE.

Además, el ID usuario debe tener los privilegios necesarios sobre el servidor DB2 para crear o grabar archivos de excepciones y de mensajes.

Sintaxis del mandato

Mandato db2se disable_autogc



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea inhabilitar la geocodificación automática.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-tableSchema *esquema_tabla*
Especifica el nombre del esquema para el valor de *esquema_tabla* especificado. Si no especifica un nombre de esquema, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

-tableName *nombre_tabla*
Especifica el nombre no calificado de la tabla que contiene la columna especificada en la que desea inhabilitar la geocodificación automática. Los desencadenantes creados para sincronizar la columna geocodificada se descartan. El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

-columnName *nombre_columna*
Especifica el nombre de la columna geocodificada que se mantiene en la que desea inhabilitar la geocodificación automática. El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

Ejemplo

El ejemplo siguiente inhabilita la geocodificación automática para una columna geocodificada denominada MYCOLUMN en la tabla MYTABLE.

```
db2se disable_autogc mydb -tableName mytable -columnName mycolumn
```

Mandato db2se disable_db

El mandato **db2se disable_db** elimina recursos que permiten a DB2 Spatial Extender almacenar y dar soporte a los datos espaciales.

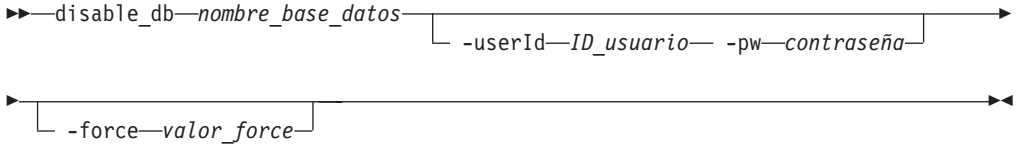
Estos recursos incluyen tipos de datos espaciales, tipos de índices espaciales, vistas de catálogo, funciones proporcionadas y procedimientos almacenados que se crearon al habilitar una base de datos para dar soporte a las operaciones espaciales.

Autorización

El ID de usuario debe tener autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se disable_db



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea inhabilitar DB2 Spatial Extender.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-force *valor_force*
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. 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).

Si especifica un valor distinto de cero para el parámetro **-force**, la base de datos se inhabilita y se eliminan todos los recursos de DB2 Spatial Extender. Si especifica cero como *valor_force*, la base de datos se inhabilita únicamente si los objetos de base de datos son dependientes de los tipos espaciales o de las funciones espaciales.

Notas sobre el uso

Si ya ha definido las columnas espaciales pero desea inhabilitar una base de datos para operaciones espaciales, debe especificar un valor distinto de 0 (cero) para el parámetro force para eliminar todos los recursos espaciales de la base de datos que no tienen otras dependencias.

Ejemplo

En el ejemplo siguiente se inhabilita el soporte para las operaciones espaciales en la base de datos *MYDB* incluso si hay objetos de base de datos con dependencias de los tipos y funciones espaciales.

```
db2se disable_db mydb -force 1
```

Mandato db2se drop_cs

El mandato **db2se drop_cs** suprime la definición de un sistema de coordenadas.

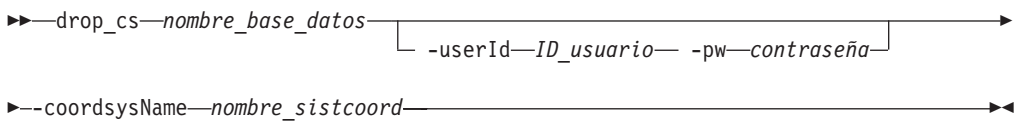
Este mandato suprime información acerca de un sistema de coordenadas de la base de datos. La información deja de estar disponible en la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Autorización

El ID de usuario debe tener autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se drop_cs



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea suprimir el sistema de coordenadas.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña del *id_usuario* especificado.

-coordsysName *nombre_sistcoord*
Identifica de forma exclusiva el sistema de coordenadas que desea suprimir.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles. La longitud máxima de este parámetro es de 128.

Notas sobre el uso

No puede descartar un sistema de coordenadas en el que se basa un sistema de referencia espacial.

Ejemplo

El ejemplo siguiente descarta un sistema de coordenadas denominado MYCOORDSYS.

```
db2se drop_cs mydb -coordsysName mycoordsys
```

Mandato db2se drop_srs

El mandato **db2se drop_srs** suprime la definición de un sistema de referencia espacial.

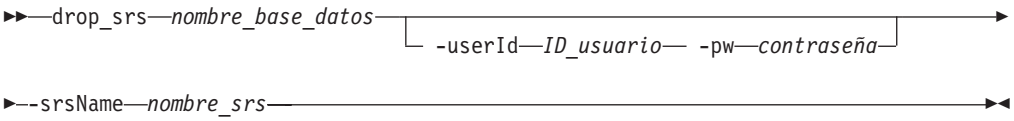
La información sobre el sistema de referencia espacial deja de estar disponible en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Autorización

El ID de usuario tiene autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se drop_srs



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea suprimir la definición de un sistema de referencia espacial.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-srsName *nombre_srs*
Identifica el sistema de referencia espacial que desea suprimir. El valor de *nombre_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

Notas sobre el uso

Si ha registrado columnas espaciales que utilizan un sistema de referencia espacial, no puede descartar dicho sistema de referencia espacial.

Ejemplo

El ejemplo siguiente descarta un sistema de referencia espacial denominado MYSRS.

```
db2se drop_srs mydb -srsName mysrs
```

Mandato db2se enable_autogc

El mandato **db2se enable_autogc** habilita la geocodificación automática.

Este mandato habilita DB2 Spatial Extender para sincronizar una columna geocodificada con las 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. La información sobre las columnas geocodificadas está disponible en la vista de catálogo DB2GSE.ST_GEOCODING.

Autorización

El ID de usuario debe tener uno de los privilegios o autorizaciones siguientes para ejecutar este mandato:

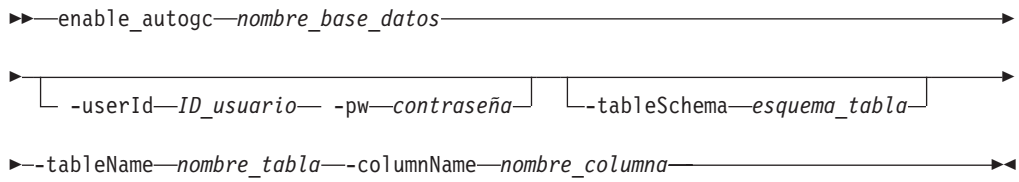
- Autorización DBADM y DATAACCESS sobre la base de datos que contiene la tabla en la que están definidos los desencadenantes que ha creado este procedimiento almacenado
- Privilegio CONTROL sobre la tabla
- Privilegio ALTER sobre la tabla

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 del mandato

Mandato db2se enable_autogc



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea habilitar la geocodificación automática.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-tableSchema *esquema_tabla*
Especifica el nombre del esquema para el valor de *esquema_tabla* especificado. Si no especifica un nombre de esquema, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

-tableName *nombre_tabla*
Especifica el nombre no calificado de la tabla que contiene la columna especificada en la que desea habilitar la geocodificación automática. Los desencadenantes se crean para sincronizar la columna geocodificada. El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

-columnName *nombre_columna*
Identifica la columna en la que se insertarán o actualizarán los datos

geocodificados. Esta columna se denomina columna geocodificada. El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

Notas sobre el uso

Antes de habilitar la geocodificación automática, debe realizar el paso de configuración de la geocodificación para especificar los valores del geocodificador y del parámetro de geocodificación. Además, identifica las columnas de geocodificación que se van a sincronizar con las columnas geocodificadas.

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.

Ejemplo

El ejemplo siguiente configura una geocodificación automática para una columna denominada MICOLUMNA en la tabla MITABLA

```
db2se enable_autogeocoding mibd -tableName mitabla -columnName micolumna
```

Mandato db2se enable_db

El mandato **db2se enable_db** proporciona a la base de datos los recursos que necesita para almacenar datos espaciales y para dar soporte a operaciones espaciales.

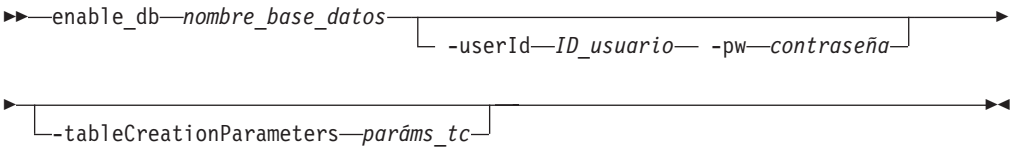
Estos recursos incluyen tipos de datos espaciales, tipos de índices espaciales, vistas de catálogo, funciones proporcionadas y otros procedimientos almacenados.

Autorización

El ID de usuario debe tener autorización DBADM, DATAACCESS y CTRLACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se enable_db



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea habilitar DB2 Spatial Extender.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-tableCreationParameters *paráms_tc*
Especifica los parámetros para las sentencias CREATE TABLE que se utilizan para crear las tablas de catálogo de DB2 Spatial Extender. Utilice la sintaxis de los parámetros para la sentencia CREATE TABLE. En el ejemplo siguiente, en el sistema operativo Windows, se especifica un espacio de tablas en el que se crearán las tablas y un espacio de tablas en el que se crearán los índices de tabla:

```
-tableCreationParameters "IN nombre_ET INDEX IN nombre_ET_ÍND"
```

La longitud máxima de este parámetro es de 32.672 caracteres.

Notas sobre el uso

Asegúrese de que dispone de un espacio de tablas temporal del sistema con un tamaño de página de 8 KB o superior y con un tamaño mínimo de 500 páginas. Se trata de un requisito para ejecutar el mandato **db2se enable_db** correctamente.

Ejemplo

El ejemplo siguiente habilita una base de datos denominada MIBD para operaciones espaciales.

```
db2se enable_db mibd
```

Mandato db2se export_shape

El mandato **db2se export_shape** exporta una columna espacial y su tabla asociada a un archivo de formas.

Puede utilizar este mandato para crear la definición de 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 el sistema de referencia espacial especificado. Las extensiones son los valores mínimos y máximos posibles de las coordenadas X, Y, Z y M. La información acerca del sistema de coordenadas está disponible en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

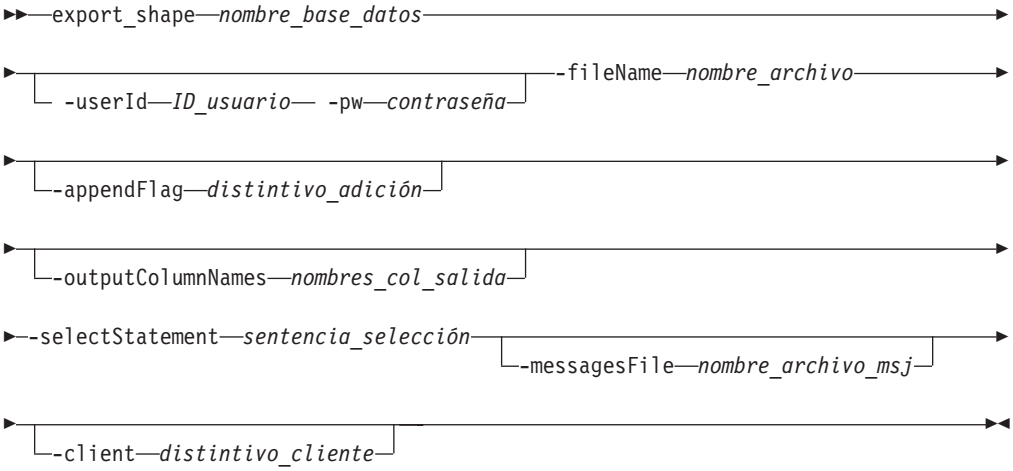
Autorización

El usuario debe tener los privilegios necesarios para ejecutar satisfactoriamente la sentencia SELECT desde la que se van a exportar los datos.

Además, el ID de propietario de la instancia de DB2 debe tener los privilegios necesarios sobre el servidor DB2 para crear o grabar archivos de formas y de mensajes.

Sintaxis del mandato

Mandato db2se export_shape



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos que contiene la tabla que se va a exportar.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-fileName *nombre_archivo*
Especifica la vía de acceso completa de un archivo de formas al que se van a exportar los datos especificados. Para ver una lista de los archivos que se graban en el servidor DB2, consulte: "Notas sobre el uso" en la página 149. La longitud máxima de este parámetro es de 256.

Si está exportando a un nuevo archivo, puede especificar la extensión opcional de archivo como .shp o .SHP. Si especifica .shp o .SHP como la extensión del archivo, DB2 Spatial Extender crea el archivo con el valor *nombre_archivo* especificado. Si no especifica la extensión de archivo opcional, DB2 Spatial Extender crea el archivo con el nombre *nombre_archivo*.shp.

Si está exportando datos añadiéndolos a un archivo existente, DB2 Spatial Extender primero busca una coincidencia exacta del nombre especificado con el parámetro **-fileName**. 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 de *distintivo_adición* indica que no está realizando adiciones a un archivo existente, pero el archivo ya existe, DB2 Spatial Extender devuelve un error y no sobregraba el archivo.

-appendFlag *distintivo_adición*
Indica si los datos que se van a exportar se van a añadir a un archivo de formas existente. Los valores posibles de este parámetro son:

- Un valor distinto de cero en *distintivo_adición* para indicar que desea añadir datos a un archivo de formas ya existente. Si la estructura del archivo existente no coincide con los datos exportados, se devuelve un error.
- Un valor de 0 en *distintivo_adición* para indicar que desea exportar a un archivo nuevo. DB2 Spatial Extender no sobregraba ningún archivo existente.

-outputColumnNames *nombres_col_salida*
Especifica uno o varios nombres de columna (separados por comas) que se utilizan para columnas no espaciales en el archivo dBASE de salida. Si este parámetro no se especifica, se utilizan los nombres de la sentencia SELECT.

Si los nombres de las columnas no están entre comillas dobles, se convierten a mayúsculas. El número de columnas especificadas debe coincidir con el número de columnas que se devuelven de la sentencia SELECT indicada en el parámetro *sentencia_selección*, excluyendo la columna espacial.

La longitud máxima de este parámetro es de 32.672 caracteres.

-selectStatement *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. La longitud máxima de este parámetro es de 32.672 caracteres.

-messagesFile *nombre_archivo_msj*
Especifica el nombre de la vía de acceso completa del archivo en el servidor DB2 en la que DB2 Spatial Extender grabará mensajes sobre la operación de exportación. Si especifica este parámetro y el archivo ya existe, se devuelve un error y la operación de exportación termina. Si no especifica este parámetro, DB2 Spatial Extender no crea un archivo de mensajes.

En el archivo de mensajes se graban los tipos de mensajes siguientes:
- 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

La longitud máxima de este parámetro es de 256.

-client *distintivo_cliente*
Especifica si la operación de exportación se lleva a cabo en el cliente o en el servidor DB2 y dónde se crean los archivos. Los valores posibles de este parámetro son:
- 0 para indicar que la operación de exportación se lleva a cabo en el servidor DB2 y que los archivos se crean en el servidor DB2.
- 1 para indicar que la operación de exportación se lleva a cabo en el cliente y que los archivos se crean en el cliente.

Si no especifica este parámetro, el valor 0 es el valor por omisión.

Notas sobre el uso

Puede exportar sólo una columna espacial cada vez.

Puede realizar el proceso de exportación en el cliente en el que se ejecuta el mandato. Normalmente es más cómodo, porque no requiere acceder al sistema de archivos del servidor DB2.

El mandato **db2se export_shape** crea o 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 20. 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

El ejemplo siguiente exporta una columna espacial denominada MICOLUMNA y su tabla asociada, MITABLA, al archivo de formas denominado `miarchivo_formas` que se encuentra en el cliente.

```
pdb2se export_shape mibd -fileName /home/micuenta/miarchivo_formas
        -selectStatement "select * from mitabla" -client 1
```

Mandato db2se import_shape

El mandato **db2se import_shape** importa un archivo de formas a una base de datos que está habilitada para operaciones espaciales.

Este mandato puede importar datos de forma y atributos a una tabla existente o a una tabla nueva.

Autorización

El ID de propietario de la instancia de DB2 debe tener los privilegios necesarios sobre el servidor DB2 para crear o grabar archivos de excepciones y de mensajes.

El ID de usuario debe cumplir requisitos adicionales sobre autorizaciones para ejecutar este mandato. Los requisitos varían en función de si está importando a una tabla existente o a una nueva tabla.

Requisitos para importar a una tabla existente
El ID de usuario debe tener uno de los privilegios o autorizaciones siguientes:

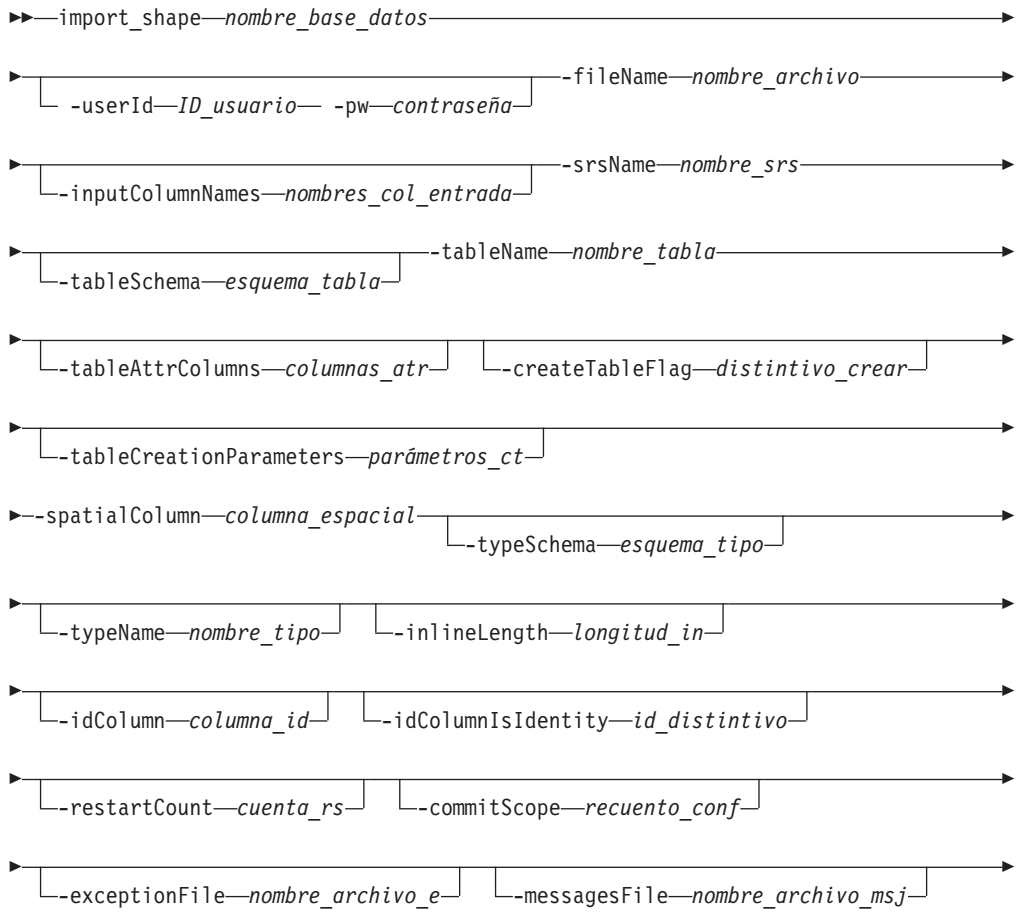
- DATAACCESS
- Privilegio CONTROL sobre la tabla o la vista
- Privilegio INSERT y SELECT en la tabla o vista

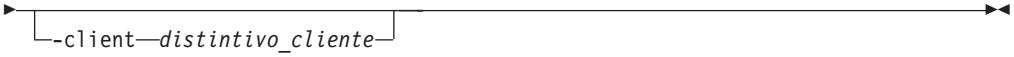
Requisitos para importar a una tabla nueva
El ID de usuario debe tener uno de los privilegios o autorizaciones siguientes:

- DBADM y DATAACCESS
- Autorización CREATETAB sobre la base de datos
- 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 del mandato

Mandato db2se import_shape





Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea importar el archivo de formas.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-fileName *nombre_archivo*
Especifica la vía de acceso completa de un archivo de formas al que se van a importar los datos especificados. Si especifica `.shp` o `.SHP` como extensión de archivo, DB2 Spatial Extender primero busca una coincidencia exacta del nombre que especifique con el parámetro **-fileName**. 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`. Para ver una lista de los archivos que se graban en el servidor DB2, consulte: "Notas sobre el uso" en la página 156.

La longitud máxima de este parámetro es de 256 caracteres.

-inputColumnNames *nombres_col_entrada*
Especifica una lista de columnas de atributos para importar desde el archivo dBASE. Si este parámetro no se especifica, se importan todas las columnas del archivo. Utilice cualquiera de los formatos siguientes para especificar una lista de atributos:

- Una lista separada por comas de los nombres de las columnas que se importarán del archivo dBASE como se muestra en el ejemplo siguiente:
  ```
  N(COLUMN1,COLUMN5,COLUMN3,COLUMN7)
  ```
  Si los nombres de las columnas no están entre comillas dobles, se convierten a mayúsculas. Los nombres resultantes deben coincidir exactamente con los nombres de columna en el archivo dBASE.
- Una lista separada por comas de los números de las columnas que se importarán del archivo dBASE como se muestra en el ejemplo siguiente:
  ```
  P(1,5,3,7)
  ```
  Las columnas están numeradas empezando por 1. Cada número de la lista debe estar separado por una coma.
- Una serie vacía `""` para indicar que no se van a importar datos de atributos.

La longitud máxima de este parámetro es de 32.672 caracteres.

-srsName *nombre_srs*
Identifica el sistema de referencia espacial (SRS) que se va a utilizar para las geometrías que se importan en la columna espacial. El valor de *nombre_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

La columna espacial no se registra. El 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 el SRS especificado. Es decir, el proceso de importación verifica que las extensiones estén dentro de las coordenadas X, Y, Z y M mínimas y máximas del SRS.

-tableSchema *esquema_tabla*
Especifica el nombre del esquema para el valor de *esquema_tabla* especificado. Si no especifica un nombre de esquema, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

-tableName *nombre_tabla*
Especifica el nombre no calificado de la tabla en la que se importará el archivo de formas. El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

-tableAttrColumns *columnas_atr*
Especifica los nombres de columna de la tabla donde se almacenarán los datos de atributos del archivo dBASE. Si este parámetro no se especifica, se utilizan los nombres de las columnas en el archivo dBASE.

El número de columnas especificadas 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. Para ver una explicación de cómo los tipos de datos de atributos se correlacionan con tipos de datos de DB2, consulte: "Notas sobre el uso" en la página 156.

Si los nombres de las columnas no están entre comillas dobles, se convierten a mayúsculas. La longitud máxima de este parámetro es de 32.672 caracteres.

-createTableFlag *distintivo_crear*
Especifica si el proceso de importación creará una tabla. Los valores posibles de este parámetro son:

- Un valor distinto de cero en *distintivo_crear* para crear una tabla. Si la tabla existe, se devuelve un error.
- Un valor de 0 en *distintivo_crear* para utilizar una tabla existente.

Si este parámetro no se especifica, se crea una tabla nueva.

-tableCreationParameters *paráms_tc*
Especifica las opciones que se añadirán a la sentencia CREATE TABLE que crea la tabla *nombre_tabla* especificada.

Para especificar las opciones CREATE TABLE, utilice la sintaxis de la sentencia CREATE TABLE. Por ejemplo, para especificar un espacio de tablas para crear tablas, índices y objetos grandes, en *paráms_tc* especifique:

```
IN Nombre_ET INDEX IN Nombre_índice_ET LONG IN Nombre_ET_largo
```

La longitud máxima de este parámetro es de 32.672 caracteres.

-spatialColumn *columna_espacial*
Especifica el nombre de la columna espacial de la tabla en la que se importarán los datos de formas.

Para una nueva tabla, este parámetro especifica el nombre de la nueva columna espacial que se creará. En el caso de una tabla que ya existe, 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.

-typeSchema *esquema_tipo*
Indica el nombre de esquema del tipo de datos espaciales especificado en *nombre_tipo*. Si este parámetro no se especifica, se utiliza DB2GSE como nombre de esquema.

El valor de *esquema_tipo* se convierte a mayúsculas a menos que esté entre comillas dobles.

-typeName *nombre_tipo*
Especifica el nombre del tipo de datos que se utilizará para los valores espaciales. Si este parámetro no se especifica, 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

Los archivos de formas, por definición, permiten la distinción sólo entre puntos y multipuntos. No hay distinción entre polígonos y multipolígonos ni entre cadenas lineales y multilíneas.

Si está importando a una tabla nueva, *nombre_tipo* 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.

-inlineLength *longitud_en_línea*
En el caso de una tabla nueva, especifica el número máximo de bytes que se asignarán para la columna espacial dentro de la tabla. Si este parámetro no se especifica, se utiliza la longitud en línea por omisión.

Los registros espaciales que exceden el tamaño indicado por *longitud_en_línea* se almacenan por separado en el espacio de tablas de LOB, al que puede ser más lento acceder.

Los tamaños típicos que son necesarios para los diversos tipos espaciales son los siguientes:

- **Un punto:** 292 bytes.
- **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 tablas para el que la tabla se ha creado.

Para obtener una descripción completa del valor de *longitud_en_línea*, consulte la sentencia CREATE TABLE en la documentación de DB2. Utilice la función de tabla ADMIN_EST_INLINE_LENGTH como ayuda para realizar una estimación de la longitud en línea necesaria para las geometrías en las tablas existentes.

-idColumn *columna_id*
Especifica el nombre de la columna que se creará para contener un número exclusivo para cada fila de datos. Los valores exclusivos para esta columna se generan automáticamente durante el proceso de importación. No puede especificar un nombre de *columna_id* que coincida con el nombre de alguna

columna en el archivo dBASE. Algunas herramientas espaciales requieren una columna con un identificador exclusivo.

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, como INTEGER, SMALLINT o BIGINT.
- En el caso de una tabla nueva, la columna se define como se indica a continuación:

  ```
  INTEGER NOT NULL PRIMARY KEY
  ```

  Si *columna_id_es_identidad* tiene el valor cero, la definición se amplía así:

  ```
  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.

-idColumnIsIdentity *columna_id_es_identidad*
Indica si la *columna_id* especificada se creará utilizando la cláusula IDENTITY. Si especifica un valor distinto de cero en *columna_id_es_identidad*, la columna *columna_id* se crea como columna de identidad. Este parámetro se ignora para tablas que ya existen.

-restartCount *recuento_reinicios*
Especifica que la operación de importación se inicia en el registro *n* + 1. Se saltan los *n* primeros registros. Si este parámetro no se especifica, se importarán todos los registros, empezando por el número 1.

-commitScope *ámbito_confirm*
Especifica que se realizará un COMMIT después de que se importen *n* registros como mínimo. Si no se especifica este parámetro, se realiza una acción COMMIT al final de la operación. Esto puede provocar un uso elevado del archivo de anotaciones cronológicas y que se pierdan datos en las operaciones que se interrumpan.

-exceptionFile *archivo_excepciones*
Especifica la vía de acceso completa de un archivo de formas en el que se graban los datos de formas que no se han podido importar. Si el parámetro no se especifica, no se crea un archivo de excepciones.

Si especifica un valor para el parámetro e incluye la extensión opcional de archivo, especifique .shp o .SHP. Si no especifica una extensión, se agrega la extensión `.shp` a *archivo_excepciones*.

El archivo de excepciones contiene el conjunto completo de filas para la sentencia de Insertar que ha fallado. Una sola sentencia de insertar puede añadir varias filas. 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, incluida la que contiene datos de formas incorrectos. Puesto que la sentencia de insertar falla, todo el conjunto de 20 filas se graba en el archivo de excepciones.

Los registros sólo se graban en el archivo de excepciones cuando dichos registros se pueden identificar correctamente, como ocurre 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 puede grabar 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 DB2. Para ver una explicación de estos archivos, consulte: "Notas sobre el uso".

Si el archivo *archivo_excepciones* ya existe, el mandato devolverá un error.

La longitud máxima de este parámetro es de 256 caracteres.

-messagesFile *nombre_archivo_msj*
Especifica el nombre de la vía de acceso completa del archivo en el servidor DB2 en la que DB2 Spatial Extender grabará mensajes sobre la operación de importación. Si no especifica este parámetro, DB2 Spatial Extender no crea un archivo de mensajes.

En el archivo de mensajes se graban los tipos de mensajes siguientes:

- 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 de error corresponden a los datos de formas que se almacenan en el archivo de excepciones *archivo_excepciones*.

Si el archivo *nombre_archivo_msj* ya existe, el mandato devolverá un error.

La longitud máxima de este parámetro es de 256 caracteres.

-client *distintivo_cliente*
Especifica si la operación de importación se lleva a cabo en el cliente o en el servidor DB2 y dónde se crean los archivos. Los valores posibles de este parámetro son:

- 0 para indicar que la operación de importación se lleva a cabo en el servidor DB2 y que se accede a los archivos desde el servidor DB2.
- 1 para indicar que la operación de importación se lleva a cabo en el cliente y que se accede a los archivos desde el cliente.

Si no especifica este parámetro, el valor 0 es el valor por omisión.

Notas sobre el uso

Puede realizar el proceso de importación en el cliente en el que se ejecuta el mandato. Normalmente es más cómodo, porque no requiere acceder al sistema de archivos del servidor DB2.

El mandato **db2se import_shape** crea o graba los cuatro archivos siguientes:

- 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 21. 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	*lon*	*dec*	DECIMAL(*lon*,*dec*)	*lon*<32
F	*lon*	*dec*	REAL	*lon* + *dec* < 7
F	*lon*	*dec*	DOUBLE	
C	*lon*		CHAR(*lon*)	
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

El mandato siguiente importa los datos del archivo de formas `miarchivo` que se encuentra en el cliente, en la tabla MITABLA. Los datos espaciales de `miarchivo` se insertan en la columna MICOLUMNA de la tabla MITABLA.

```
db2se import_shape mibd -fileName miarchivo -srsName NAD83_SRS_1
        -tableName mitabla -spatialColumnName micolumna -client 1
```

Mandato db2se register_gc

El mandato **`db2se register_gc`** registra un geocodificador.

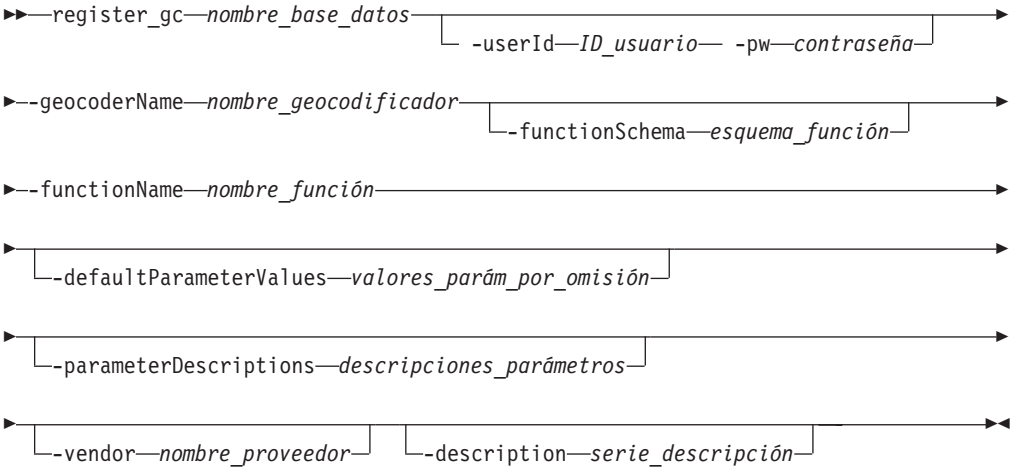
Este mandato registra un geocodificador para que puede utilizarse para geocodificar valores en una tabla y almacenar o actualizar el valor de geometría resultante. La información sobre los geocodificadores registrados se encuentra en la vista de catálogo DB2GSE.ST_GEOCODERS.

Autorización

El ID de usuario debe tener autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se register_gc



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea registrar un geocodificador.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-geocoderName *nombre_geocodificador*
Identifica de forma exclusiva el geocodificador que desea registrar. El valor de *nombre_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles. La longitud máxima de este parámetro es de 128 caracteres.

-functionSchema *esquema_función*
Especifica el nombre del esquema para la función que implementa este geocodificador. Si este parámetro no se especifica, se utiliza el valor del registro especial CURRENT SCHEMA como el nombre de esquema para la función.

El valor *esquema_función* se convierte a mayúsculas a menos que esté entre comillas dobles.

-functionName *nombre_función*
Especifica el nombre no calificado de la función que implementa este geocodificador. La función ya debe estar creada y listada en la vista de catálogo SYSCAT.ROUTINES.

El valor de *nombre_función*, junto con el valor implícita o explícitamente definido de *esquema_función*, debe identificar de forma exclusiva la función.

El valor de *nombre_función* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

-defaultParameterValues *valores_parám_por_omisión*
Especifica la lista de valores por omisión de parámetros de configuración para la función de geocodificador.

Debe especificarlos valores de parámetro en el orden en que la función los ha definido y separarlos mediante una coma. Por ejemplo:

```
valor_parm1_omisión,valor_parm2_omisión,...
```

Cada valor de parámetro debe ser una expresión de SQL. Siga estas directrices para especificar los valores de parámetro por omisión:

- 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, especifique la expresión siguiente para indicar NULL en un parámetro de tipo entero:

  ```
  CAST(NULL AS INTEGER)
  ```
- Si el parámetro de geocodificación va a ser una columna de geocodificación, no especifique el valor por omisión del parámetro.

Utilice dos comas consecutivas (`...,,...`) para omitir los valores por omisión de los parámetros que indique al configurar la geocodificación o al ejecutar la geocodificación en modalidad de proceso por lotes mediante el parámetro **-parameterValues** con el mandato **db2se setup_gc** o el mandato **db2se run_gc**.

La longitud máxima de este parámetro es de 32.672 caracteres.

-parameterDescriptions *descripciones_parámetros*
Especifica la lista de descripciones de parámetros de configuración para la función de geocodificador. La longitud máxima de este parámetro es de 32.672 caracteres.

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 va a utilizar una coma dentro de la descripción de un parámetro, ponga la serie de caracteres entre comillas simples o dobles. Por ejemplo:

```
descripción,'descripción2, que contiene una coma',descripción3
```

-vendor *nombre_proveedor*
Especifica el nombre del proveedor que ha implementado el geocodificador. La longitud máxima de este parámetro es de 128 caracteres.

-description *serie_descripción*
Describe el geocodificador explicando su aplicación. La longitud máxima de este parámetro es de 256 caracteres.

Notas sobre el uso

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 de tipo de datos de 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.

Ejemplo

El ejemplo siguiente registra un geocodificador denominado "mygeocoder", que es utilizado por una función denominada "myschema.myfunction".

```
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"
```

Mandato db2se register_spatial_column

El mandato **db2se register_spatial_column** registra una columna espacial y le asocia un sistema de referencia espacial (SRS).

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.

También puede utilizar este mandato para actualizar la información de extensión espacial.

La información sobre las extensiones espaciales y las columnas espaciales registradas se encuentra en la vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS.

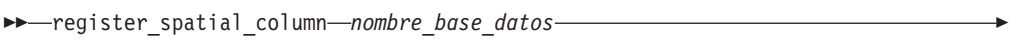
Autorización

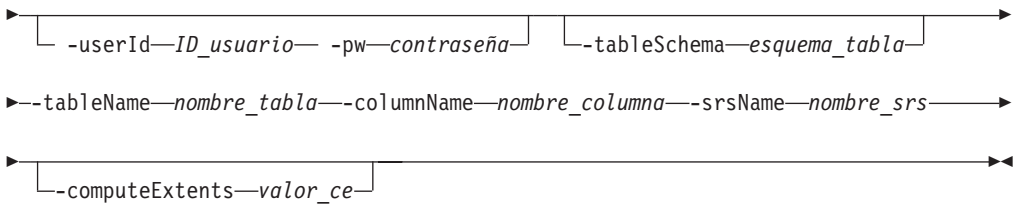
El ID de usuario debe tener uno de los privilegios o autorizaciones siguientes para ejecutar este mandato:

- Autorización DBADM y DATAACCESS 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 del mandato

Mandato db2se register_spatial_column





Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea registrar una columna espacial.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-tableSchema *esquema_tabla*
Especifica el nombre del esquema para el valor de *esquema_tabla* especificado. Si no especifica un nombre de esquema, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

-tableName *nombre_tabla*
Especifica el nombre no calificado de la tabla que contiene la columna que se está registrando. El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

-columnName *nombre_columna*
Identifica la columna que se va a registrar. El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

-srsName *nombre_srs*
Identifica el sistema de referencia espacial que se va a utilizar para esta columna espacial. El valor de *nombre_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

computeExtents *valor_ce*
Indica si se calcularán las extensiones geográficas de una columna especificada y si se incluirán en la vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS. Los valores posibles de este parámetro son:

- Un valor mayor que 0 para calcular las extensiones geográficas.
- Nulo, 0 o un valor negativo para impedir este cálculo.

Si omite este parámetro, el efecto será el mismo que especificar 0. El cálculo de las extensiones no se llevará a cabo.

Ejemplo

El ejemplo siguiente registra una columna espacial denominada MYCOLUMN en la tabla MYTABLE, con el sistema de referencia espacial "USA_SRS_1".

```
db2se register_spatial_column mydb -tableName mytable -columnName mycolumn -srsName USA_SRS_1
```

Mandato db2se remove_gc_setup

El mandato **db2se remove_gc_setup** elimina toda la información de configuración de geocodificación correspondiente a una columna geocodificada.

La información que está asociada con la columna geocodificada especificada deja de estar disponible en las vistas de catálogo DB2GSE.ST_GEOCODING y DB2GSE.ST_GEOCODING_PARAMETERS.

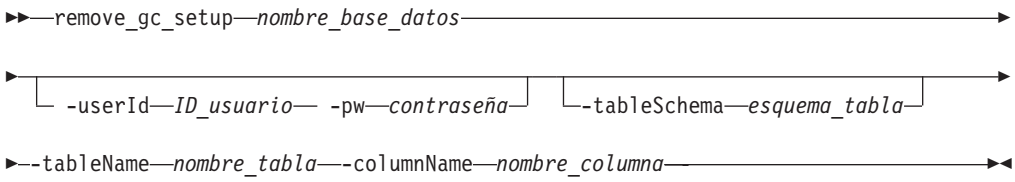
Autorización

El ID de usuario debe tener uno de los privilegios o autorizaciones siguientes para ejecutar este mandato:

- 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 del mandato

Mandato db2se remove_gc_setup



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea eliminar toda la información de configuración de geocodificación correspondiente a una columna geocodificada.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-tableSchema *esquema_tabla*
Especifica el nombre del esquema para el valor de *esquema_tabla* especificado. Si no especifica un nombre de esquema, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

-tableName *nombre_tabla*
Especifica el nombre no calificado de la tabla para el *nombre_columna* que se ha especificado. El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

-columnName *nombre_columna*
Identifica el nombre de la columna de la que desea eliminar la configuración

de geocodificación. El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

Notas sobre el uso

No puede eliminar una configuración de geocodificación si la geocodificación automática está habilitada para la columna geocodificada.

Ejemplo

El ejemplo siguiente elimina una configuración para las operaciones de geocodificación que se aplican a una columna espacial denominada MICOLUMNA de la tabla MITABLA.

```
db2se remove_gc_setup mibd -tableName mitabla-columnName micolumna
```

Mandato db2se restore_indexes

El mandato **db2se restore_indexes** restaura los índices espaciales que ha guardado previamente emitiendo el mandato **db2se save_indexes** a una base de datos habilitada para operaciones espaciales.

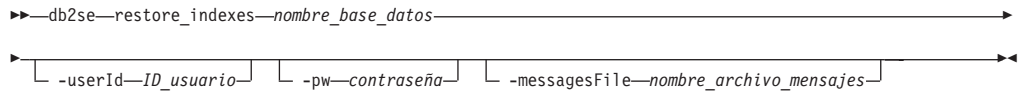
Este mandato se utiliza para volver a crear los índices espaciales al actualizar una instancia de 32 bits a 64 bits.

Autorización

Autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales.

Sintaxis del mandato

Mandato db2se restore_indexes



Parámetros del mandato

nombre_base_datos
Nombre de la base de datos que desea actualizar.

-userId ID_usuario
ID de usuario de base de datos que tiene autorización SYSADM o DBADM para la base de datos que se desea actualizar.

-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

El mandato db2se save_indexes guarda los índices espaciales definidos en una base de datos habilitada para operaciones espaciales.

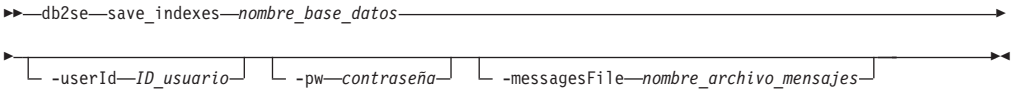
Este mandato se utiliza para guardar las definiciones de índices espaciales actuales al actualizar una instancia de 32 bits a 64 bits.

Autorización

Autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales.

Sintaxis del mandato

Mandato db2se save_indexes



Parámetros del mandato

nombre_base_datos
Nombre de la base de datos que desea actualizar.

-userId ID_usuario
ID de usuario de base de datos que tiene autorización SYSADM o DBADM para la base de datos que se desea actualizar.

-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 run_gc

El mandato **db2se run_gc** ejecuta un geocodificador en modalidad de proceso por lotes en una columna geocodificada.

La información que está asociada con la columna geocodificada especificada deja de estar disponible en las vistas de catálogo DB2GSE.ST_GEOCODING y DB2GSE.ST_GEOCODING_PARAMETERS.

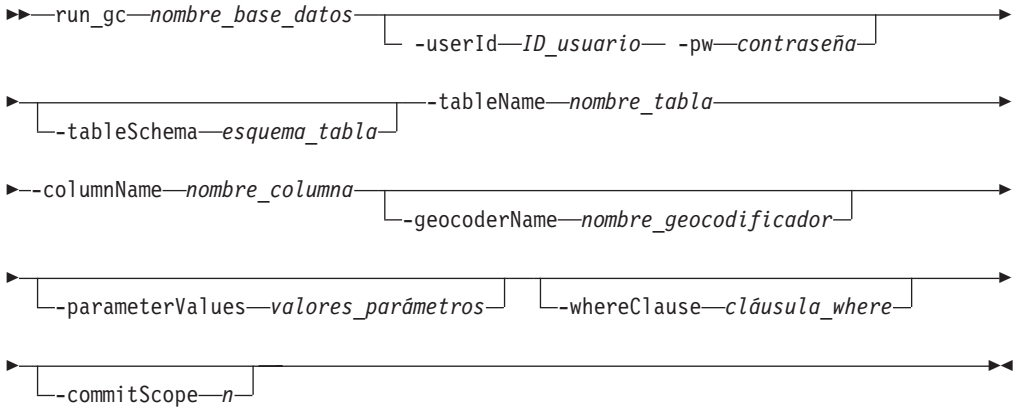
Autorización

El ID de usuario debe tener uno de los privilegios o autorizaciones siguientes para ejecutar este mandato:

- 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 del mandato

Mandato db2se run_gc



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea ejecutar un geocodificador en modalidad de proceso por lotes en una columna geocodificada.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-tableSchema *esquema_tabla*
Especifica el nombre del esquema para el valor de *esquema_tabla* especificado. Si no especifica un nombre de esquema, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

-tableName *nombre_tabla*
Especifica el nombre no calificado de la tabla para el *nombre_columna* que se ha especificado. El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

-columnName *nombre_columna*
Identifica el nombre de la columna en la que se insertarán o actualizarán los datos geocodificados. El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

-geocoderName *nombre_geocodificador*
Identifica de forma exclusiva el geocodificador que va a realizar la geocodificación. El valor de *nombre_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles. La longitud máxima de este parámetro es de 128 caracteres.

-parameterValues *valores_parámetros*
Especifica la lista de valores de parámetros de configuración para la función de geocodificador. Si este parámetro no se especifica, 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 que se especificaron al registrar el geocodificador.

Debe especificarlos valores de parámetro en el orden en que la función los ha definido y separarlos mediante una coma. Por ejemplo:

valor_parm1_omisión,*valor_parm2_omisión*,...

Cada valor de parámetro debe ser una expresión de SQL. Siga estas directrices para especificar los valores de parámetro por omisión:

- 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, especifique la expresión siguiente para indicar NULL en un parámetro de tipo entero: `CAST(NULL AS INTEGER)`.
- Si el parámetro de geocodificación va a ser una columna de geocodificación, no especifique el valor por omisión del parámetro.

Utilice dos comas consecutivas (`...,,...`) para omitir los valores de los parámetros en los que indicó un valor al configurar o registrar el geocodificador.

La longitud máxima de este parámetro es de 32.672 caracteres.

-whereClause *cláusula_where*
Especifica el texto para una condición de búsqueda de una cláusula WHERE para filtrar el conjunto de registros que se geocodificarán. Si este parámetro no se especifica, se utiliza el valor de *cláusula_where* especificado durante la configuración de la geocodificación. Si no se ha especificado un valor para *cláusula_where* al configurar la geocodificación, se geocodifican todas las filas de la tabla.

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 en *cláusula_where*.

La longitud máxima de este parámetro es de 32.672 caracteres.

-commitScope *n*
Especifica que se a va realizar un COMMIT después de geocodificar *n* registros. Si este parámetro no se especifica, se utiliza el valor especificado con el parámetro **-commitScope** durante la configuración de la geocodificación. Si no se especifica este parámetro ni tampoco un valor durante la configuración de la geocodificación, se realiza una acción COMMIT al final de la operación. Una acción COMMIT al final de la operación puede provocar un uso elevado del archivo de anotaciones cronológicas y que se pierdan datos en las operaciones que se interrumpan.

Ejemplo

El ejemplo siguiente ejecuta un geocodificador en modalidad de proceso por lotes para llenar una columna denominada MYCOLUMN de una tabla denominada MYTABLE.

```
db2se run_gc mydb -tableName mytable -columnName mycolumn
```

Mandato db2se setup_gc

El mandato **db2se setup_gc** asocia una columna que se va a geocodificar con un geocodificador y configura los parámetros de geocodificación correspondientes.

La información acerca de esta configuración está disponible en las vistas de catálogo DB2GSE.ST_GEOCODING y DB2GSE.ST_GEOCODING_PARAMETERS.

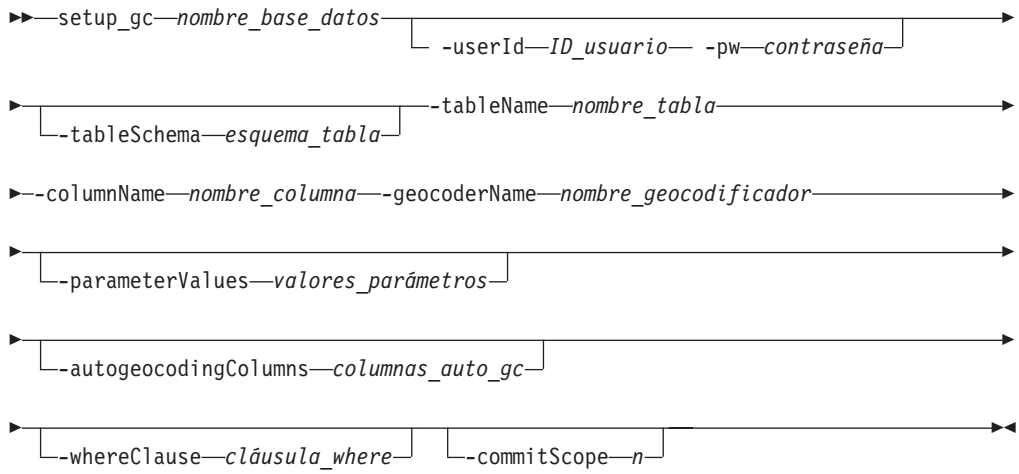
Autorización

El ID de usuario debe tener uno de los privilegios o autorizaciones siguientes para ejecutar este mandato:

- 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 del mandato

Mandato db2se setup_gc



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea configurar un geocodificador.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-tableSchema *esquema_tabla*
Especifica el nombre del esquema para el valor de *esquema_tabla* especificado. Si no especifica un nombre de esquema, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

-tableName *nombre_tabla*
Especifica el nombre no calificado de la tabla para el *nombre_columna* que se ha especificado. El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

-columnName *nombre_columna*
Identifica el nombre de la columna en la que se insertarán o actualizarán los datos geocodificados. El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

-geocoderName *nombre_geocodificador*
Identifica de forma exclusiva un geocodificador ya registrado que va a realizar la geocodificación. El valor de *nombre_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles. La longitud máxima de este parámetro es de 128 caracteres.

-parameterValues *valores_parámetros*
Especifica la lista de valores de parámetros de configuración para la función de geocodificador. Si este parámetro no se especifica, se utilizan los valores de parámetros por omisión que se especificaron al registrar el geocodificador.

Debe especificarlos valores de parámetro en el orden en que la función los ha definido y separarlos mediante una coma. Por ejemplo:

```
valor_parm1_omisión,valor_parm2_omisión,...
```

Cada valor de parámetro debe ser una expresión de SQL. Siga estas directrices para especificar los valores de parámetro por omisión:

- 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, especifique la expresión siguiente para indicar NULL en un parámetro de tipo entero: `CAST(NULL AS INTEGER)`.
- Si el parámetro de geocodificación va a ser una columna de geocodificación, no especifique el valor por omisión del parámetro.

Utilice dos comas consecutivas (`...,,...`) para omitir los valores de los parámetros que indicó al configurar o registrar el geocodificador.

La longitud máxima de este parámetro es de 32.672 caracteres.

-whereClause *cláusula_where*
Especifica el texto para una condición de búsqueda de una cláusula WHERE para filtrar el conjunto de registros que se geocodificarán. Si este parámetro no se especifica, se geocodifican todas las filas de la tabla.

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 en *cláusula_where*.

La longitud máxima de este parámetro es de 32.672 caracteres.

-commitScope *n*
Especifica que se a va realizar un COMMIT después de geocodificar *n* registros. Si no se especifica este parámetro, se realiza una acción COMMIT al final de la operación. Una acción COMMIT al final de la operación puede provocar un uso elevado del archivo de anotaciones cronológicas y que se pierdan datos en las operaciones que se interrumpan.

Notas sobre el uso

Este mandato 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. Los valores de los parámetros que se especifican en la configuración de la geocodificación alteran temporalmente cualquiera de los valores por omisión de los parámetros que se han especificado en el registro del geocodificador.

Debe ejecutar este mandato antes de habilitar la geocodificación automática. La configuración de los parámetros de geocodificación debe realizarse antes de habilitar la geocodificación automática.

Puede ejecutar este mandato antes de la geocodificación en modalidad de proceso por lotes. Si no especifica valores de parámetros para ejecutar la geocodificación en modalidad de proceso por lotes, se utilizan los valores de parámetros especificados en la configuración de la geocodificación. Si especifica valores de parámetros, estos valores alterarán temporalmente los especificados en la configuración de la geocodificación.

Ejemplo

El ejemplo siguiente configura operaciones de geocodificación para llenar una columna espacial denominada MICOLUMNA en la tabla MITABLA.

```
db2se setup_gc mibd -tableName mitabla -columnName micolumna
        -parameterValues "address,city,state,zip,2,90,70,20,1.1,'meter',4.."
        -autogeocodingColumns address,city,state,zip
        -commitScope 10
```

Mandato db2se shape_info

El mandato **db2se shape_info** muestra información acerca de un archivo de formas y su contenido. En el caso de una base de datos especificada, este mandato puede, opcionalmente, mostrar todos los sistemas de coordenadas y de referencia espacial compatibles que están incluidos en el archivo de formas.

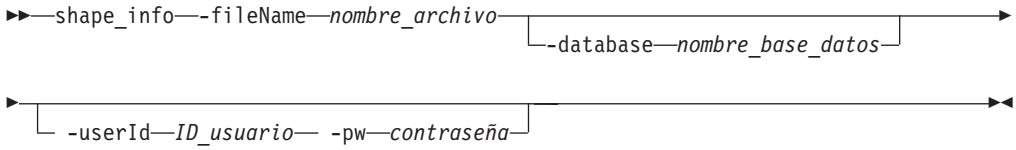
Autorización

El ID de usuario debe tener los privilegios necesarios sobre el servidor DB2 para leer los archivos de formas.

Si se especifica el parámetro **-database**, el ID de usuario debe tener privilegio SELECT sobre la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Sintaxis del mandato

Mandato db2se shape_info



Parámetros del mandato

Donde:

-fileName *nombre_archivo*
Especifica el nombre de la vía de acceso completa del archivo de formas cuya información desea visualizar.

DB2 Spatial Extender primero busca una coincidencia exacta del nombre que especifique con el parámetro **-fileName**. 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`.

La longitud máxima de este parámetro es de 256 caracteres.

-database *nombre_base_datos*
Especifica el nombre de la base de datos para la que desea buscar todos los sistemas de coordenadas y de referencia espacial compatibles incluidos en el archivo de formas *nombre_archivo*.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

Ejemplo

El ejemplo siguiente muestra información acerca del archivo de formas denominado `MYFILE`, que reside en el directorio actual.

```
db2se shape_info -fileName myfile
```

El ejemplo siguiente muestra información acerca de un archivo de formas de ejemplo para UNIX denominado `offices`. El parámetro *–database* busca todos los sistemas de referencia espacial y los sistemas de coordenadas compatibles en la base de datos definida (en este caso, MIBD).

```
db2se shape_info -fileName ~/sqllib/samples/extenders/spatial/data/offices -database miBD
```

Mandato db2se unregister_gc

El mandato **db2se unregister_gc** deshace el registro de un geocodificador.

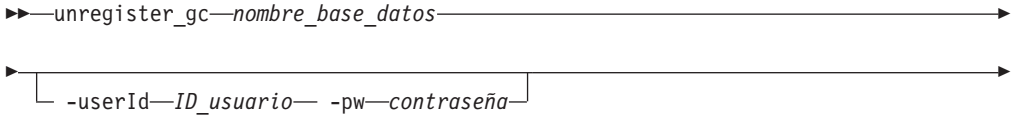
Para buscar información sobre el geocodificador del que desea deshacer el registro, consulte la vista de catálogo DB2GSE.ST_GEOCODERS.

Autorización

El ID de usuario debe tener autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales para ejecutar este mandato.

Sintaxis del mandato

Mandato db2se unregister_gc



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea deshacer el registro de un geocodificador.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-geocoderName *nombre_geocodificador*
Identifica de forma exclusiva el geocodificador cuyo registro desea deshacer. El valor de *nombre_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles. La longitud máxima de este parámetro es de 128 caracteres.

Notas sobre el uso

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, comprueba las vista de catálogo DB2GSE.ST_GEOCODING y DB2GSE.ST_GEOCODING_PARAMETERS.

Ejemplo

El ejemplo siguiente cancela el registro de un geocodificador denominado MYGEOCODER.

```
db2se unregister_gc mydb -geocoderName mygeocoder
```

Mandato db2se unregister_spatial_column

El mandato **db2se unregister_spatial_column** elimina el registro de una columna espacial.

Este mandato elimina el registro haciendo lo siguiente:

- Eliminando la asociación del sistema de referencia espacial con la columna espacial. La vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS sigue mostrando la columna espacial, pero la columna ya no está asociada con ningún sistema de referencia espacial.
- Para una tabla base, descartando la restricción creada en esta tabla para garantizar que todos los valores de la geometría en esta columna espacial estén representados en el mismo sistema de referencia espacial.

Autorización

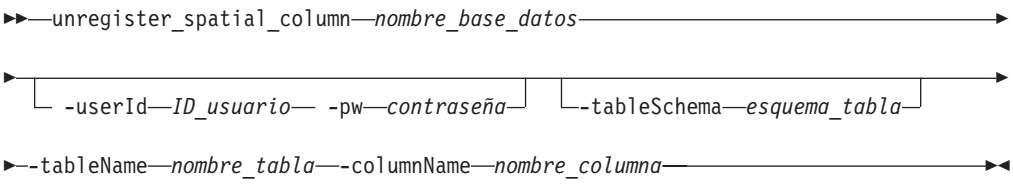
El ID de usuario debe tener uno de los privilegios o autorizaciones siguientes para ejecutar este mandato:

- Autorización DBADM y DATAACCESS 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 del mandato

Mandato db2se unregister_spatial_column



Parámetros del mandato

Donde:

nombre_base_datos
Especifica el nombre de la base de datos para la que desea eliminar el registro de una columna espacial.

-userId *ID_usuario*
Especifica el ID de usuario de base de datos que tiene autorización DATAACCESS sobre la base de datos indicada por *nombre_base_datos*.

-pw *contraseña*
Especifica la contraseña de *id_usuario*.

-tableSchema *esquema_tabla*
Especifica el nombre del esquema para el valor de *esquema_tabla* especificado. Si no especifica un nombre de esquema, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

-tableName *nombre_tabla*
Especifica el nombre no calificado de la tabla para el *nombre_columna* que se ha especificado. El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

-columnName *nombre_columna*
Identifica la columna cuyo registro se eliminará. El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

Ejemplo

El ejemplo siguiente cancela el registro de una columna espacial denominada mycolumn en la tabla mytable.

```
db2se unregister_spatial_column mydb -tableName mytable -columnName mycolumn
```

Mandato db2se upgrade

El mandato **db2se upgrade** actualiza una base de datos habilitada para operaciones espaciales de la Versión 9.5 o 9.7 a la Versión 10.1.

Este mandato puede descartar 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.

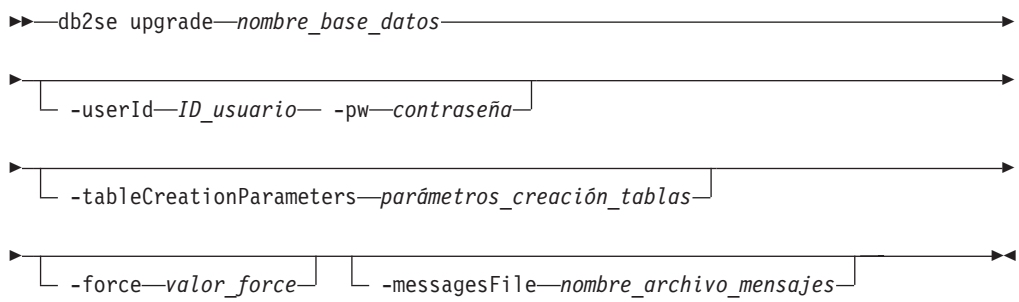
Consejo: Ejecute el mandato **db2se upgrade** con la opción `-force 0` y especifique un archivo de mensajes para determinar qué índices se deben actualizar sin realizar un proceso de actualización adicional.

Autorización

Autorización DBADM y DATAACCESS sobre la base de datos habilitada para operaciones espaciales que desea actualizar.

Sintaxis del mandato

Mandato db2se upgrade



Parámetros del mandato

Donde:

nombre_base_datos
Nombre de la base de datos que desea actualizar.

-userId ID_usuario
ID de usuario de base de datos que tiene la autorización DATAACCESS sobre la base de datos que se está actualizando.

-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_force

- 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 nombre_archivo_mensajes
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** comprueba varias condiciones y devuelve uno o más de los siguientes errores si algunas de estas condiciones no son ciertas:

- 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 superior y con un tamaño mínimo de 500 páginas. Se trata de un requisito para ejecutar el mandato **db2se upgrade** correctamente.

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 descartar 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.

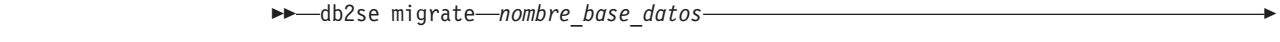
Consejo: Ejecute el mandato **db2se migrate** con la opción `-force 0` y especifique un archivo de mensajes para determinar qué índices se deben migrar sin realizar un proceso de migración adicional.

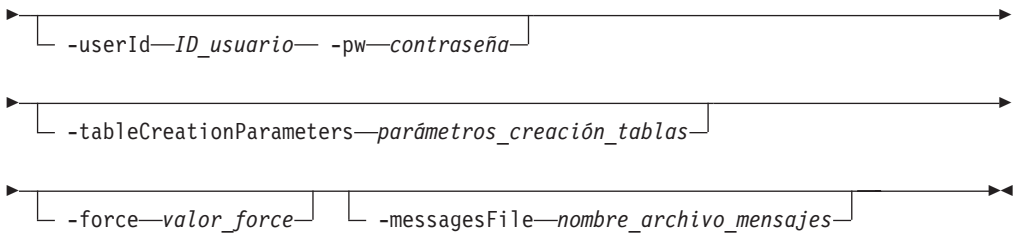
Autorización

Autorización SYSADM o DBADM sobre la base de datos espacial que desea migrar.

Sintaxis del mandato

Mandato db2se migrate





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.

Capítulo 17. Procedimientos almacenados

Utilice los procedimientos almacenados de DB2 Spatial Extender para configurar DB2 Spatial Extender y crear proyectos que utilicen datos espaciales.

Cuando configura DB2 Spatial Extender o el procesador de línea de mandatos de DB2, invoca implícitamente estos procedimientos almacenados. Por ejemplo, cuando emita el mandato de CLP **db2se create_srs**, se llamará al procedimiento DB2GSE.ST_CREATE_SRS.

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 procedimientos almacenados de DB2 Spatial Extender en una base de datos, realice 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 superior y con un tamaño mínimo de 500 páginas. Éste es un requisito para poder ejecutar correctamente el procedimiento almacenado ST_ENABLE_DB o el mandato **db2se enable_db**.
2. Habilite la base de datos para las operaciones espaciales invocando el procedimiento ST_ENABLE_DB. Consulte “Procedimiento ST_ENABLE_DB” en la página 200 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:

© Copyright IBM Corp. 1998, 2012

Las implementaciones de los procedimientos almacenados se archivan en la biblioteca db2gse en el servidor de DB2 Spatial Extender.

Procedimiento ST_ALTER_COORDSYS

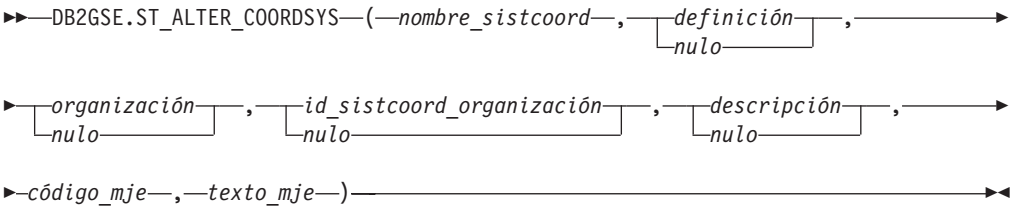
Utilice este procedimiento almacenado para actualizar una definición de sistema de coordenadas en la base de datos. Cuando se procesa este sistema almacenado, la información sobre el sistema de coordenadas se actualiza en la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

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



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.

Procedimiento 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.

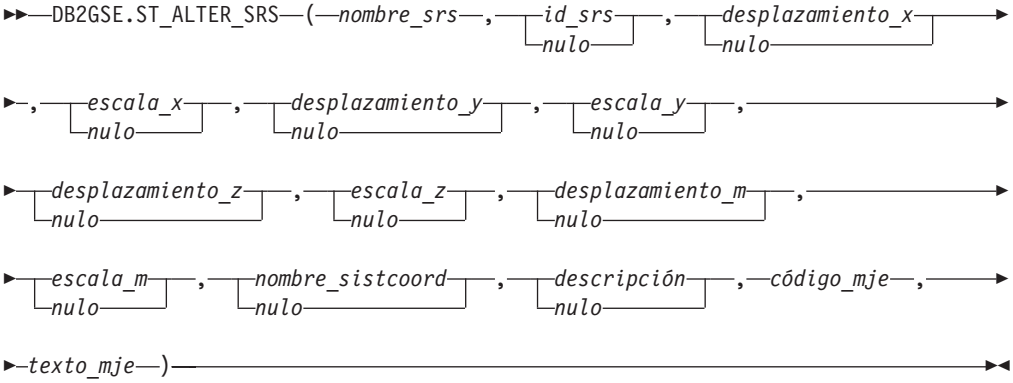
Restricción: No puede modificar un sistema de referencia si una columna espacial registrada utiliza dicho sistema de referencia espacial.

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



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,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.

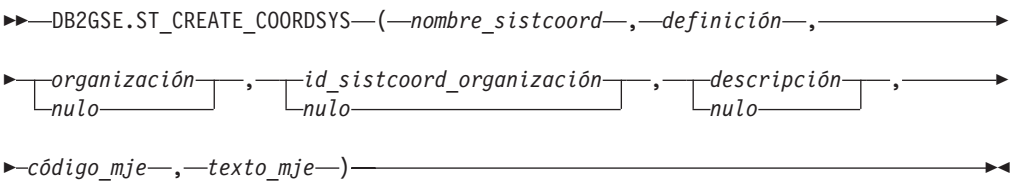
Procedimiento ST_CREATE_COORDSYS

Utilice este procedimiento almacenado para crear un 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



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.

Procedimiento ST_CREATE_SRS

Utilice el procedimiento almacenado ST_CREATE_SRS 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 mínimos y máximos posibles de 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:

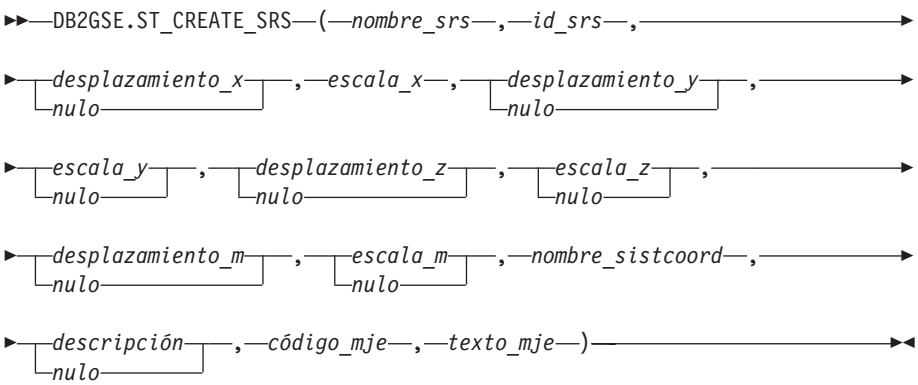
- La primera variación toma los factores de conversión (desplazamientos y factores de escala) como parámetros de entrada.
- La segunda variación toma las extensiones y la precisión como parámetros de entrada y calcula internamente los factores de conversión.

Autorización

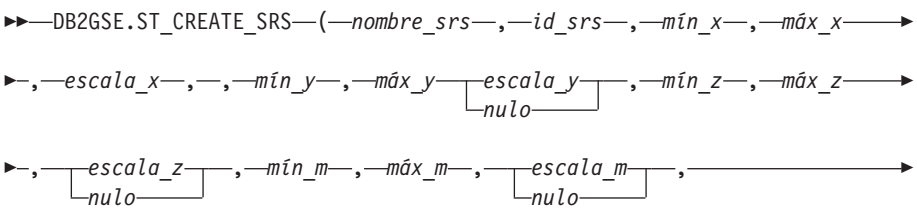
No se necesita ninguna.

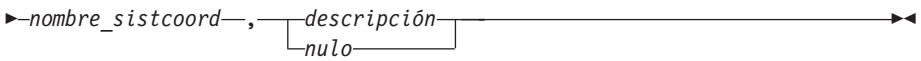
Sintaxis

Con factores de conversión (versión 1)



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.

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 mínimo 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.

Procedimiento 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.

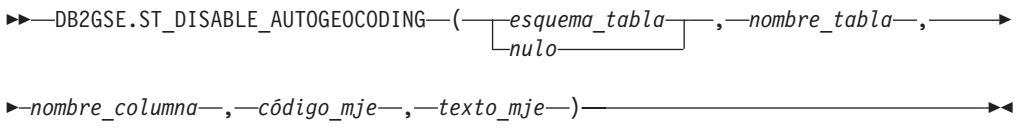
Autorización

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



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.

Procedimiento ST_DISABLE_DB

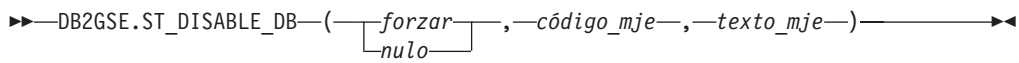
Utilice este procedimiento almacenado para eliminar recursos que permitan a DB2 Spatial Extender almacenar y dar soporte a datos espaciales.

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



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.

Procedimiento ST_DROP_COORDSYS

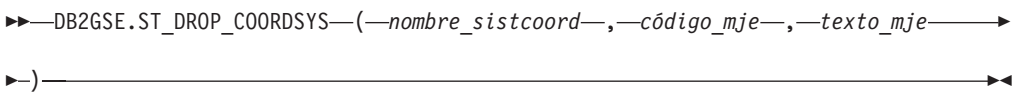
Utilice este procedimiento almacenado para suprimir información acerca de un sistema de coordenadas de la base de datos. Cuando se procesa este procedimiento almacenado, la información sobre el sistema de coordenadas dejará de estar disponible en la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

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



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.

Procedimiento 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.

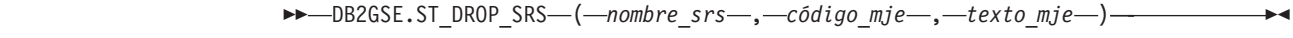
Restricción: No puede descartar un sistema de referencia espacial si una columna espacial que utiliza dicho sistema de referencia espacial está registrado.

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



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.

Procedimiento 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 llamando 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

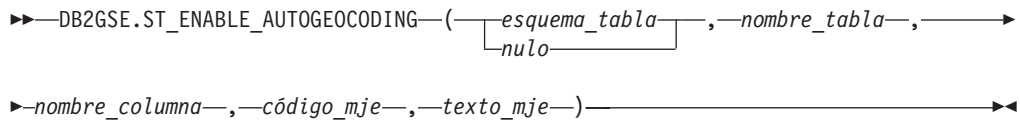
El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las autorizaciones o uno de los privilegios siguientes:

- 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



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.

Procedimiento ST_ENABLE_DB

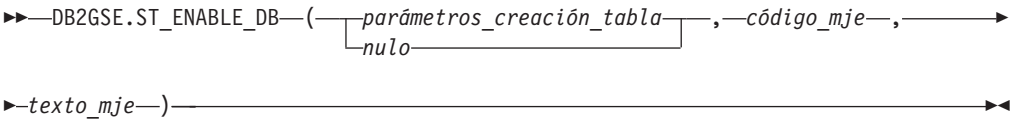
Utilice este procedimiento almacenado para proporcionar a la base de datos los recursos que necesita para almacenar datos espaciales y para dar soporte a operaciones espaciales. Estos recursos incluyen tipos de datos espaciales, tipos de índices espaciales, vistas de catálogo, funciones proporcionadas 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



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 tablas 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 contexto 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);
```

Procedimiento ST_EXPORT_SHAPE

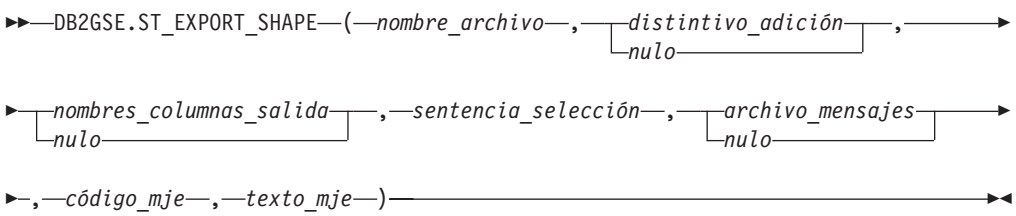
Utilice este procedimiento almacenado para exportar una columna espacial y su tabla asociada a un archivo de formas.

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



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 de formas y escribir en los archivos.

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 o 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 22. 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.

Procedimiento 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.

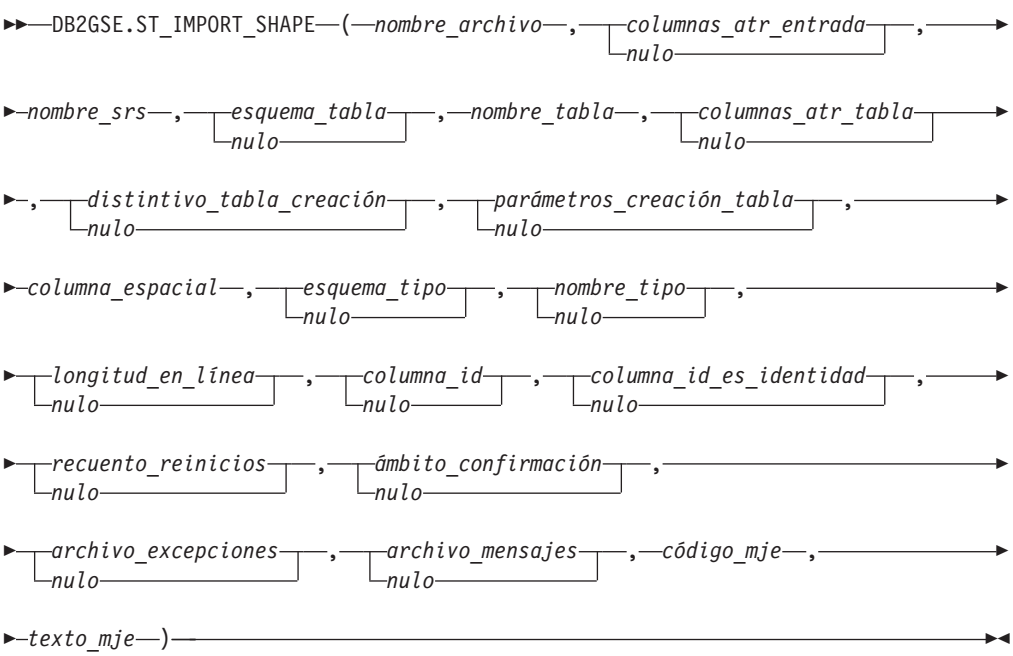
Autorización

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



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 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 tablas para crear tablas, especifique:

```
IN Nombre_ET INDEX IN Nombre_índice_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 tablas 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 tablas 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 se realizará una acción COMMIT al final de la operación. Esto puede provocar un uso elevado del archivo de anotaciones cronológicas y que se pierdan datos en las operaciones que se interrumpan.

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 23. 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	*lon*	*dec*	DECIMAL(*lon*,*dec*)	*lon*<32
F	*lon*	*dec*	REAL	*lon* + *dec* < 7
F	*lon*	*dec*	DOUBLE	
C	*lon*		CHAR(*lon*)	
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,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.

Procedimiento ST_REGISTER_GEOCODER

Utilice este procedimiento almacenado para registrar un geocodificador.

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 de los parámetros utilizados para llamar al procedimiento 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.

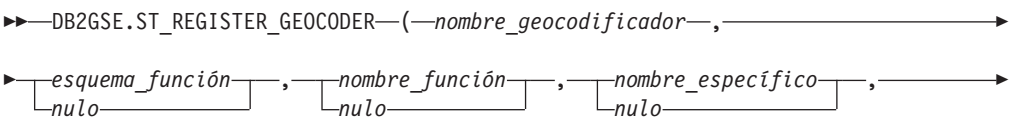
El tipo de retorno de la función de geocodificador sirve como el tipo de datos para la columna geocodificada. El tipo de retorno puede ser cualquier tipo de datos de DB2, tipo definido por el usuario o tipo estructurado. Si se devuelve un tipo definido por el usuario o un tipo estructurado, la función 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.

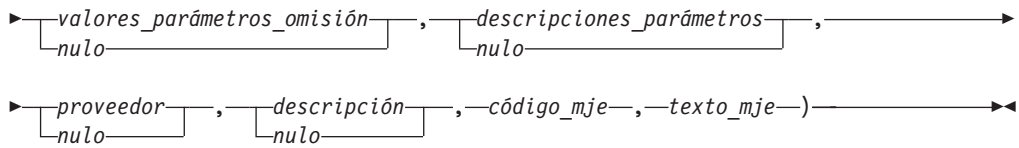
Para saber si un geocodificador ya está registrado, consulte la vista de catálogo DB2GSE.ST_GEOCODERS.

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





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.

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.

Procedimiento ST_REGISTER_SPATIAL_COLUMN

Utilice este procedimiento almacenado para registrar una columna espacial y para asociar un sistema de referencia espacial (SRS) al mismo.

También puede utilizar este procedimiento para calcular las extensiones geográficas de la columna espacial.

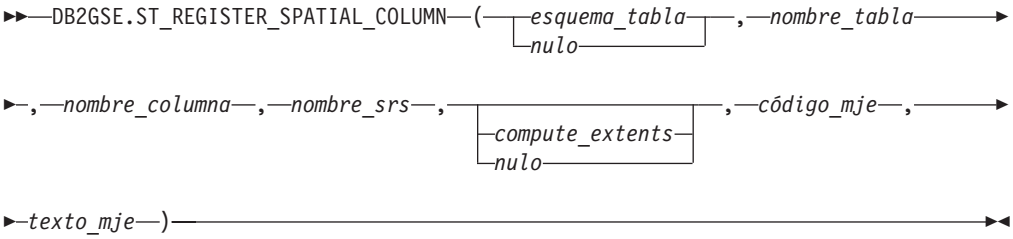
Después de procesar este procedimiento almacenado, la información sobre la columna espacial registrada y las extensiones geográficas está disponible en la vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS. Si se registra una columna espacial, se crea una restricción en la tabla, si es posible, para asegurar 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



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).

compute_extents
Indica si se calcularán las extensiones geográficas de una columna especificada y si se incluirán en la vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS. Los valores posibles de este parámetro son:

- Un valor mayor que 0 para calcular las extensiones geográficas.
- Nulo, 0 o un valor negativo para impedir este cálculo.

Si omite este parámetro, el efecto será el mismo que especificar 0. El cálculo de las extensiones no se llevará a cabo.

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_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.

Procedimiento 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.

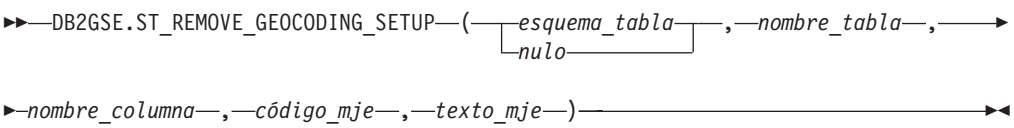
Restricción:
No puede eliminar una configuración de geocodificación si la geocodificación automática está habilitada para la columna geocodificada.

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 las que operará el geocodificador especificado
- Privilegio CONTROL sobre esta tabla
- Privilegio UPDATE sobre esta tabla

Sintaxis



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.

Procedimiento ST_RUN_GEOCODING

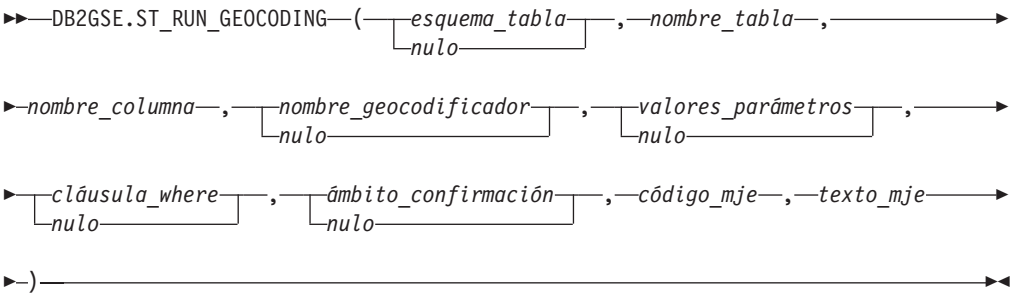
Utilice este procedimiento almacenado para ejecutar un geocodificador en modalidad de proceso por lotes en una columna geocodificada.

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



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 geocodificador. 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 a va 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 se realizará una acción COMMIT al final de la operación.

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 SAMPLEGC y el valor de 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',
     'SAMPLEGC',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.

Procedimiento 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 acerca de la configuración está disponible 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 ST_RUN_GEOCODING en modalidad de proceso por lotes.

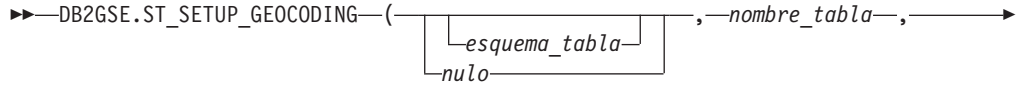
Este paso es un requisito necesario para la geocodificación automática. No puede habilitar la geocodificación automática sin configurar primero los parámetros de geocodificación. Este paso es un requisito necesario para la geocodificación de proceso por lotes. Puede ejecutar la geocodificación en modalidad 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.

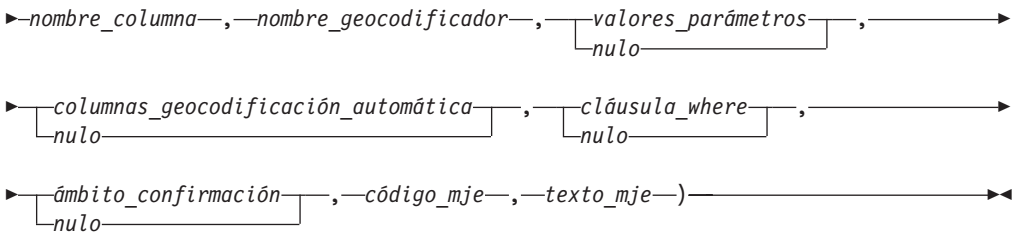
Autorización

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 las que operará el geocodificador especificado
- Privilegio CONTROL sobre esta tabla
- Privilegio UPDATE sobre esta tabla

Sintaxis





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 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).

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 SAMPLEGC:

```
call DB2GSE.ST_SETUP_GEOCODING(NULL, 'CUSTOMERS', 'LOCATION',
    'SAMPLEGC','ADDRESS,CITY,STATE,ZIP,1,100,80,,,,"$HOME/sqllib/
     gse/refdata/ky.edg","$HOME/sqllib/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.

Procedimiento ST_UNREGISTER_GEOCODER

Utilice este procedimiento almacenado para eliminar el registro de un geocodificador.

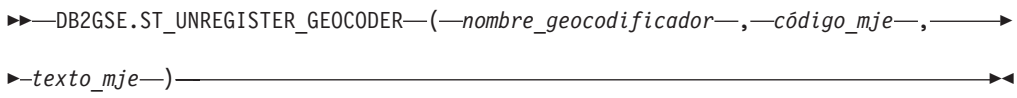
Restricción:
No puede deshacer el registro de un geocodificador si está especificado en la configuración de geocodificación para cualquier columna.

Para determinar si un geocodificador está especificado en la configuración de geocodificación para una columna, comprueba 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



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.

Procedimiento 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:

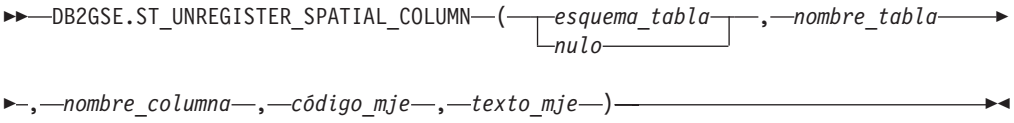
- Eliminando la asociación del sistema de referencia espacial con la columna espacial. La vista de catálogo ST_GEOMETRY_COLUMNS sigue conteniendo la columna espacial, pero la columna ya no está asociada con ningún sistema de referencia espacial.
- Para una tabla base, descartando la restricción que DB2 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



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 18. Funciones espaciales

Utilice funciones espaciales para programar sus aplicaciones. Conozca los factores que son comunes para todas o para la mayoría de las funciones espaciales y el uso de funciones por categoría.

Consideraciones sobre las funciones espaciales 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
  ```

© Copyright IBM Corp. 1998, 2012

Las funciones ST_BuildMBRAggr y ST_BuildUnionAggr se describen en "MBR Aggregate" (Agregado MBR) y "Union Aggregate" (Agregado de unión), respectivamente.

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 Spatial Extender 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 Spatial Extender no ha encontrado ninguna función llamada ST_PointOnSurface 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 no devuelve un valor ST_MultiPolygon, DB2 Spatial Extender emite una excepción de ejecución.

Si una función devuelve una geometría de un supertipo, normalmente el operador TREAT puede indicar a DB2 Spatial Extender 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 Spatial Extender 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

Lista de 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 24 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 24. 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 24. 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 24. 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

Categorías y usos de las funciones espaciales

En esta sección se presentan todas las funciones espaciales, organizadas por categoría.

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 de constructor para la conversión de formatos de intercambio de datos

DB2 Spatial Extender proporciona funciones espaciales que convierten geometrías entre formatos de intercambio de datos.

Los formatos de intercambio de datos que reciben soporte son:

- 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 convierten formatos de intercambio de datos en geometrías

La conversión de las representaciones geométricas de los formatos de intercambio de datos en valores de geometría permite el intercambio de datos en forma de valores de geometría.

Funciones que convierten geometrías en representaciones de texto convencional (WKT)

La conversión de geometrías en representaciones WKT permite el intercambio de geometrías en formato de texto ASCII. Las representaciones WKT son valores CLOB que representan series de caracteres 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 convierten 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**

Funciones que convierten geometrías en representaciones binarias convencionales (WKB)

La conversión de geometrías en representaciones WKB permite el intercambio de geometrías en formato binario. 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**

Funciones que convierten geometrías en representaciones de forma ESRI

La conversión de geometrías en representaciones de forma ESRI permite el intercambio de geometrías en formato binario. 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.

Funciones que convierten geometrías en representaciones GML (Geography Markup Language)

La conversión de geometrías en representaciones GML permite el intercambio de geometrías en formato de texto ASCII. Las representaciones GML son series 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.

Función que crea geometrías a partir de coordenadas

La función ST_Point crea geometrías no solo a partir de formatos de intercambio de datos, sino también a partir de valores de coordenadas. Esta función es muy útil si los datos de ubicación ya están almacenados en la base de datos.

Ejemplos de llamadas a las funciones de constructor

Consulte los ejemplos de código para llamar a funciones de constructor, crear tablas donde guardar los datos resultantes de las funciones de constructor y recuperar los datos resultantes para aprender a utilizar las funciones de constructor.

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();
```

Funciones de comparación de 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. En la siguiente tabla se indican las funciones de comparación y su finalidad.

Tabla 25. Funciones de comparación por el objetivo

Finalidad	Funciones
Determinar si el interior de una geometría forma intersección con el interior de otra.	• ST_Contains • ST_Within
Devolver información sobre las intersecciones de geometrías.	• ST_Crosses • ST_Intersects • ST_Overlaps • ST_Touches
Determinar si el rectángulo más pequeño que engloba a una geometría forma intersección con el rectángulo más pequeño que engloba a otra geometría.	• ST_EnvIntersects • ST_MBRIntersects
Determinar si dos objetos son idénticos.	• ST_Equals • ST_EqualCoordsys • ST_EqualSRS
Determina si las geometrías comparadas cumplen las condiciones de la serie de la matriz patrón DE-9IM.	• ST_Relate
Comprueba si hay una intersección entre dos geometrías.	• ST_Disjoint

En DB2 Spatial Extender, las funciones de comparación devuelven un valor de 1 (uno) si una comparación cumple determinados criterios, un valor de 0 (cero) si una comparación no cumple 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; dim = 0, 1 ó 2.

F No debe existir una intersección; dim = -1.

***** No importa si existe o no una intersección; 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.

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 Osstrom), 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 que determinan 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.

Funciones que determinan si las geometrías forman intersección

Las funciones ST_Intersects, ST_Crosses, ST_Overlaps y ST_Touches determinan si una geometría interseca otra.

Estas funciones difieren principalmente respecto al ámbito de intersección para el que realizan la prueba:

- 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.

Funciones que comparan envolturas de geometrías

ST_EnvIntersects y ST_MBRIntersects son funciones similares que determinan si el rectángulo más pequeño que incluye una geometría forma intersección con el rectángulo más pequeño que incluye 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.

Funciones que comprueban si dos elementos son idénticos

Estas funciones comparan sistemas de referencia espacial, definiciones de sistema de coordenadas o geometrías.

- ST_EqualCoordsys
- ST_Equals
- ST_EqualSRS

Función que determina si las geometrías no forman intersección

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.

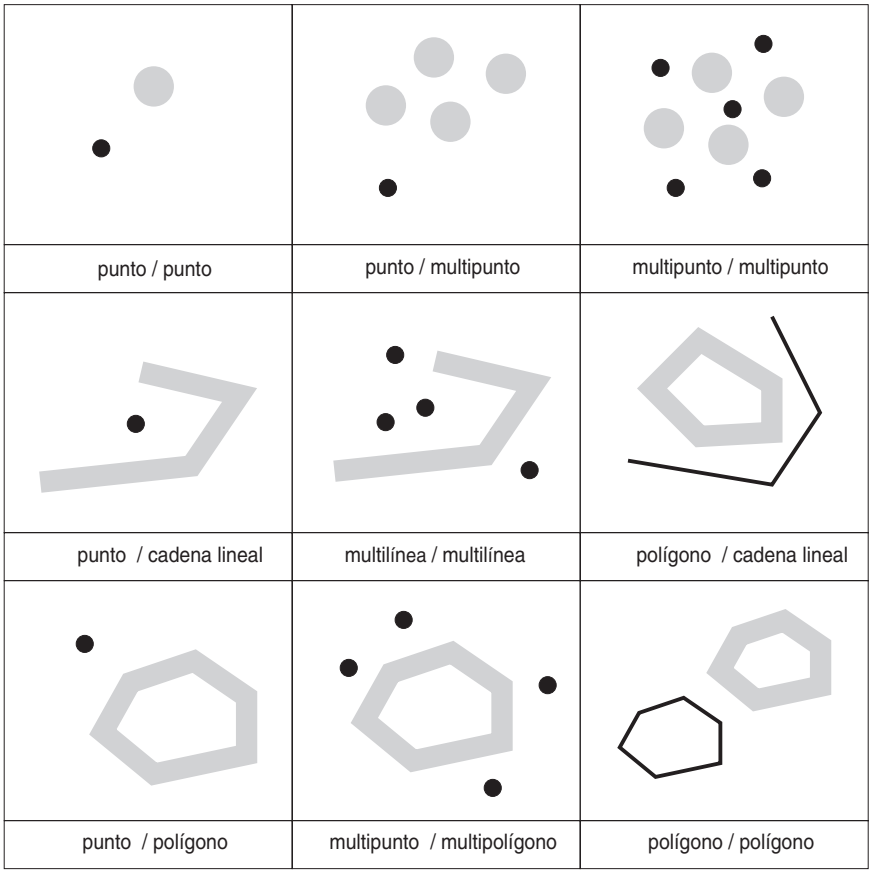


Figura 17. ST_Disjoint. Las geometrías en negro representan la geometría a; las geometrías en gris representan la geometría b. En todos los casos, la geometría a y la geometría b están inconexas la una respecto a la otra.

La matriz siguiente simplemente indica que ni los interiores ni los perímetros de ninguna geometría forman intersección.

Tabla 26. Matriz de ST_Disjoint

	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 dos 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 para obtener información sobre las geometrías y los índices

DB2 Spatial Extender proporciona funciones que devuelven información acerca de las propiedades de las geometrías y los índices espaciales.

La información que se devuelve 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
- Distancia entre geometrías
- Parámetros utilizados para definir un índice espacial o un índice en una columna 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 la categoría “Funciones espaciales que generan nuevas geometrías”.

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

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

Funciones que devuelven 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

Funciones que devuelven información sobre las dimensiones de una geometría

Las funciones siguientes devuelven información sobre la dimensión de una geometría, como el área cubierta por una geometría dada o la longitud de una curva o una multicurva concreta.

Estas funciones son:

- ST_Area
- ST_Dimension
- ST_Length

Funciones que indican si una geometría es cerrada, vacía o simple

DB2 Spatial Extender proporciona funciones que indican si una geometría es cerrada, vacía o simple.

Estas funciones indican:

- 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)

Estas funciones son:

- ST_IsClosed
- ST_IsEmpty
- ST_IsSimple

Funciones que identifican el sistema de referencia espacial asociado a 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.

Estas funciones son:

- ST_SrsId (también llamada ST_SRID)
- ST_SrsName

Función que devuelve la información sobre distancias entre geometrías

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 18 en la página 251 muestra esta información.

La figura muestra un mapa de Estados Unidos con una línea recta entre dos puntos denominados Los Angeles y Chicago.

Figura 18. 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_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 parámetro identificado por el número.

Funciones para generar 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 con un supertipo al subtipo correspondiente

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_ToGeomColl
- ST_ToLineString
- ST_ToMultiLine

- ST_ToMultiPoint
- ST_ToMultiPolygon
- ST_ToPoint
- ST_ToPolygon

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

Funciones que obtienen una geometría nueva a partir de varias

Las funciones siguientes obtienen geometrías individuales a partir de varias geometrías. Por ejemplo, la combinación de dos geometrías en una geometría individual.

- MBR Aggregate (Agregado MBR)
- ST_Union
- Union Aggregate (Agregado de unión)

Funciones que crean una geometría nueva en función de las medidas de una geometría existente

Estas funciones pueden crear una geometría nueva en función de las medidas de una geometría existente. También pueden devolver la distancia hasta una ubicación en la geometría.

Estas funciones son:

- ST_DistanceToPoint
- ST_FindMeasure o ST_LocateAlong
- ST_MeasureBetween o ST_LocateBetween
- ST_PointAtDistance

Funciones que crean formas modificadas de geometrías existentes

Las funciones siguientes crean formas modificadas de geometrías existentes, como, por ejemplo, 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

Funciones que convierten geometrías 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.

Función EnvelopesIntersect

Utilice la función EnvelopesIntersect para determinar si dos geometrías forman intersección o si una geometría forma intersección con una envoltura definida por cuatro valores de tipo DOUBLE.

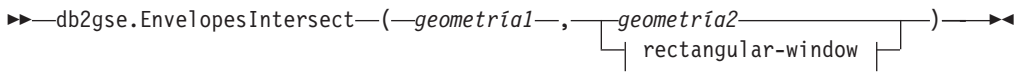
EnvelopesIntersect acepta dos tipos de parámetros de entrada:

- Dos geometrías

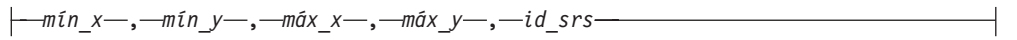
  EnvelopesIntersect devuelve un 1 si la envoltura de la primera geometría forma una intersección con la envoltura de la segunda geometría. En caso contrario, se devuelve un 0 (cero).
- Una geometría, cuatro valores de coordenadas de tipo DOUBLE que definen los ángulos inferior izquierdo y superior derecho 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



rectangular-window:



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.

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.

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.

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.

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
```

Funciones MBR Aggregate (Agregado MBR)

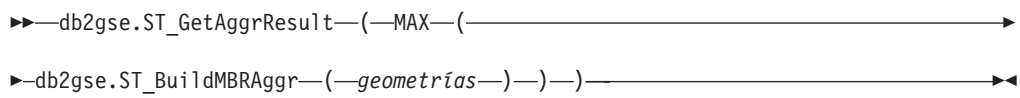
Utilice las funciones ST_BuildMBRAggr y ST_GetAggrResult juntas para agregar un conjunto de geometrías de una columna a una geometría individual. La combinación crea un rectángulo que representa el rectángulo delimitador mínimo (minimum bounding rectangle, MBR) que encierra todas las geometrías contenidas en la columna. Las coordenadas Z y M se descartan cuando se calcula el agregado.

La expresión siguiente es un ejemplo que utiliza la función MAX con la función espacial db2gse.ST_BuildMBRAggr para calcular el rectángulo delimitador mínimo de las geometrías en la columna columnName y la función espacial db2gse.ST_GetAggrResult para devolver la geometría resultante calculada para el rectángulo delimitador mínimo:

```
db2gse.ST_Get_AggrResult(MAX(db2gse.ST_BuildMBRAggr(columnName)))
```

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



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 base de datos particionada

- 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))
```

Función ST_AppendPoint

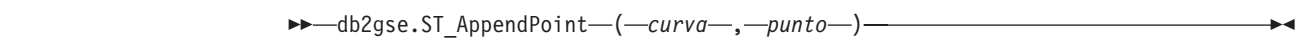
La función ST_AppendPoint toma una curva y un punto como parámetros de entrada y extiende la curva por el punto determinado. Si la curva proporcionada tiene coordenadas Z o M, el punto también debe tener coordenadas Z o M. La curva resultante se representa según el sistema de referencia espacial de la 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



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 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 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)
```

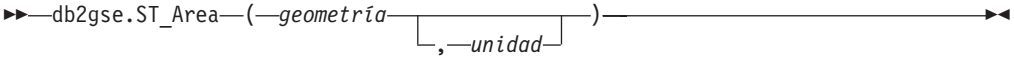
Función ST_Area

La función 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 o en la unidad por omisión.

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



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, la unidad angular asociada a este sistema de coordenadas es el valor por omisión.

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 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 +1.00000000000000E+002   +9.29034116132748E+000   +3.58702077598427E-006
 2 +2.00000000000000E+002   +1.85806823226550E+001   +7.17404155196855E-006
 3 +2.50000000000000E+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  SQUARE FEET           SQUARE METERS          SQUARE MILES
-- --------------------- --------------------- ---------------------
 1 +7.94698788000000E+008   +7.38302286101346E+007   +2.85060106320552E+001
```

Función ST_AsBinary

La función 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

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'0101000000000000000000244000000000000034400'
```

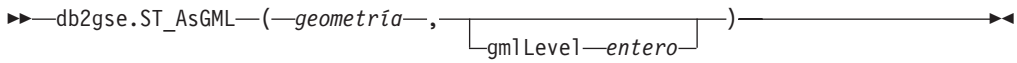
Función ST_AsGML

La función 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



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:Point srsName="EPSG:4269">     <gml:Point srsName="EPSG:4269
                                                                                    srsDimension="2">
            <gml:coord>                         <gml:coordinates>                   <gml:pos>
            <gml:X>10</gml:X><gml:Y>20</gml:Y> 10,20                                10,20
            </gml:coord></gml:Point>            </gml:coordinates></gml:Point>      </gml:pos></gml:Point>
```

Función ST_AsShape

La función 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

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'0100000000000000000002440000000000003440'
```

Función ST_AsText

La función 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

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))
```

Función ST_Boundary

La función 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.

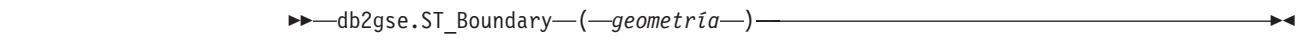
Si la geometría proporcionada es un punto, un multipunto, una curva cerrada o una multicurva cerrada, o si está vacía, el resultado es una geometría vacía de tipo ST_Point. Paras las curvas o multicurvas que no son cerradas, los puntos de inicio y final de las curvas se proporcionan en forma de valor 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



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
```

Función ST_Buffer

La función 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.

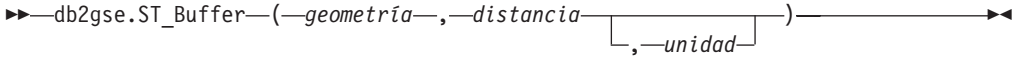
Cada punto del perímetro de la geometría resultante está separado de la geometría proporcionada por la distancia especificada. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

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



Parámetro

geometría
Valor de tipo ST_Geometry o de uno de sus subtipos que representa la geometría para la creación de la zona de separación situada alrededor.

distancia
Valor de tipo DOUBLE PRECISION que especifica la distancia que se debe utilizar para la zona de separación que debe haber alrededor de la *geometría*.

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 la *geometría* está en un sistema de coordenadas geográficas, la unidad angular asociada a este sistema de coordenadas es el valor por omisión.

Restricciones para las conversiones de unidades: Se devuelve un error (SQLSTATE 38SU4) si se produce cualquiera 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 es un valor ST_Point , 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,
   50.00000000 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))
```

Funciones MBR Aggregate (Agregado MBR)

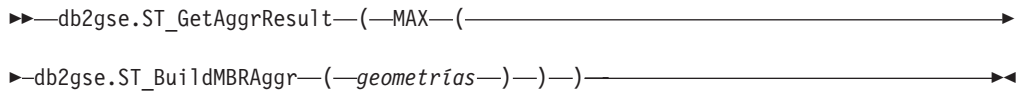
Utilice las funciones ST_BuildMBRAggr y ST_GetAggrResult juntas para agregar un conjunto de geometrías de una columna a una geometría individual. La combinación crea un rectángulo que representa el rectángulo delimitador mínimo (minimum bounding rectangle, MBR) que encierra todas las geometrías contenidas en la columna. Las coordenadas Z y M se descartan cuando se calcula el agregado.

La expresión siguiente es un ejemplo que utiliza la función MAX con la función espacial db2gse.ST_BuildMBRAggr para calcular el rectángulo delimitador mínimo de las geometrías en la columna columnName y la función espacial db2gse.ST_GetAggrResult para devolver la geometría resultante calculada para el rectángulo delimitador mínimo:

```
db2gse.ST_Get_AggrResult(MAX(db2gse.ST_BuildMBRAggr(columnName)))
```

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



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 base de datos particionada
- 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))
```

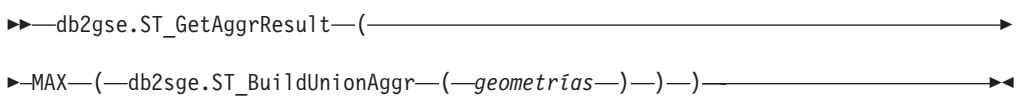
Funciones de agregado de unión

Un agregado de unión es la combinación de las funciones ST_BuildUnionAggr y ST_GetAggrResult. Utilice esta combinación para agregar 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



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 bases de datos particionadas
- 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)
```

Función ST_Centroid

La función 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

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)
```

Función ST_ChangePoint

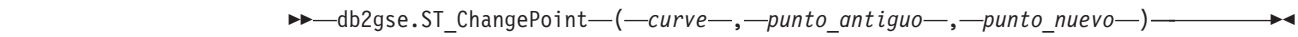
La función ST_ChangePoint toma una curva y dos puntos como parámetros de entrada. Sustituye todas las apariciones del primer punto de la curva de entrada por el segundo punto y devuelve la curva resultante. La geometría resultante se representa según el sistema de referencia espacial de la geometría 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



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
```

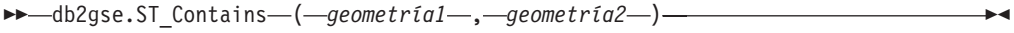
Resultados:

```
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)
```

Función ST_Contains

Utilice la función ST_Contains para determinar si una geometría está completamente dentro de otra.

Sintaxis



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*.

Tipo de retorno

INTEGER

Uso

ST_Contains toma dos geometrías como parámetros de entrada y devuelve un 1 si la primera geometría contiene completamente a la segunda o la segunda geometría está completamente contenida en la primera. De lo contrario devuelve un 0 (cero) para indicar que la primera geometría no contiene completamente a la segunda.

La función ST_Contains devuelve el resultado opuesto exacto de la función ST_Within.

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 y utiliza el sistema de referencia subyacente, se convertirá al otro sistema de referencia espacial.

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 27. 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	*

Ejemplos

La Figura 19 en la página 276 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.

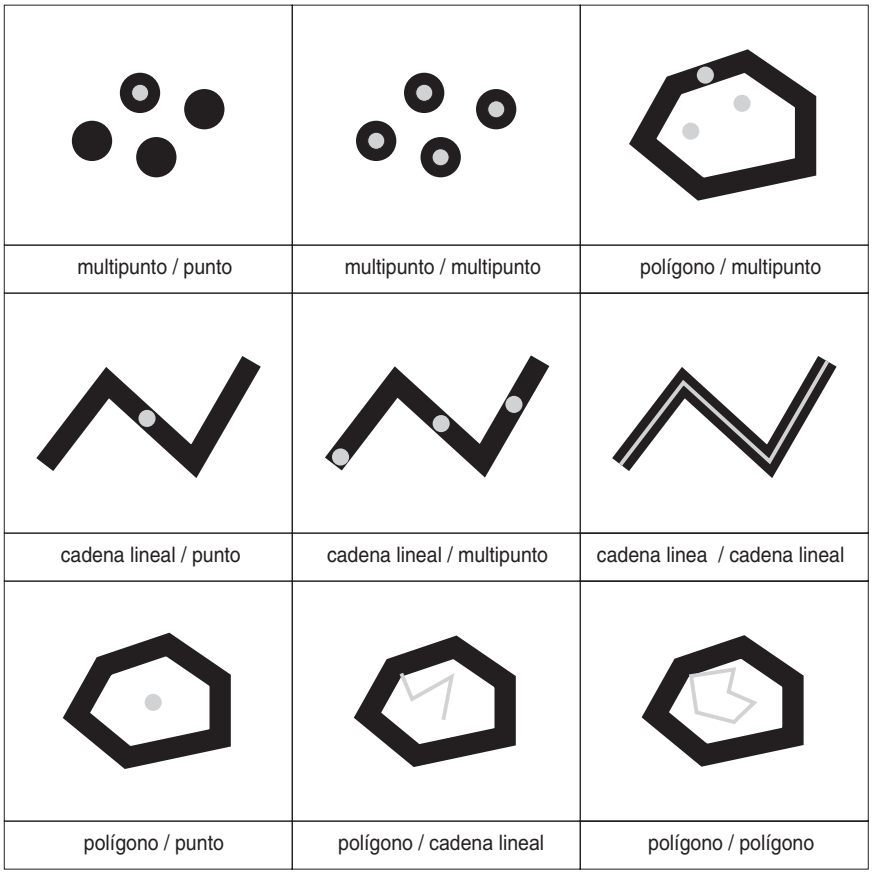


Figura 19. ST_Contains. Las geometrías en negro representan la geometría a y las geometrías en gris representan la geometría b. En todos los casos, la geometría a contiene completamente a la geometría b.

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
```

Función ST_ConvexHull

La función ST_ConvexHull utiliza una geometría como parámetro de entrada y devuelve su envoltura convexa.

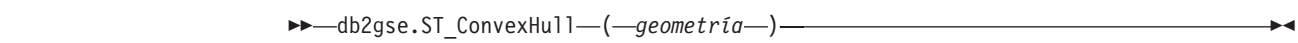
La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

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



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))
```

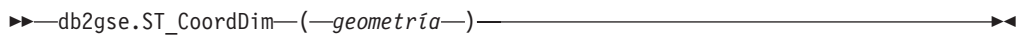
Función ST_CoordDim

La función 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



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
```

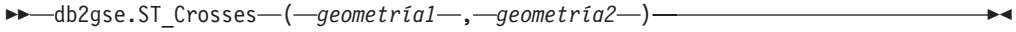
Función ST_Crosses

La función 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



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
```

Función ST_Difference

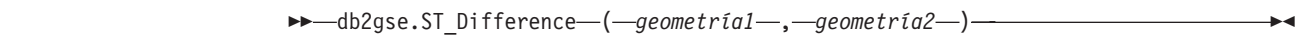
La función ST_Difference toma dos geometrías como parámetros de entrada y devuelve la parte de la primera geometría que no forma intersección con la segunda.

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.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría y utiliza el sistema de referencia subyacente, se convertirá al otro sistema de referencia espacial.

También se puede invocar esta función como método.

Sintaxis



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*.

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)
```

Función ST_Dimension

La función 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

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
```

Función ST_Disjoint

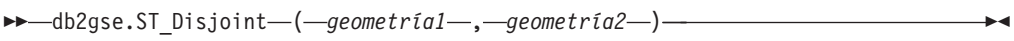
La función ST_Disjoint toma dos geometrías como parámetros de entrada y devuelve un 1 si dichas geometrías no forman intersección. Si las geometrías forman intersección, se devuelve un 0 (cero).

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



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
```

función ST_Distance

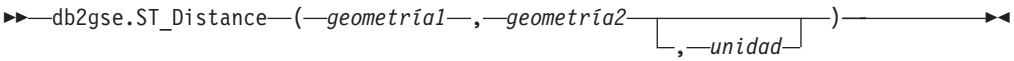
La función ST_Distance toma como parámetros de entrada dos geometrías y, opcionalmente, una unidad, y devuelve la distancia más corta entre cualquier punto de la primera geometría y cualquier punto de la segunda geometría, expresada en las unidades proporcionadas o en las unidades por omisión.

Si cualquiera de las 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 y utiliza el sistema de referencia subyacente, se convertirá al otro sistema de referencia espacial.

También puede llamar esta función como método cuando suministre una unidad de medida.

Sintaxis



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.

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, la unidad angular asociada a este sistema de coordenadas es el valor por omisión.

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.

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_GEOMTRIES1 y SAMPLE_GEOMTRIES2.

```
SELECT   sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
         sg2.id AS sg1_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 sg1_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_GEOMTRIES1 y SAMPLE_GEOMTRIES2.
SELECT   sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
         sg2.id AS sg1_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:

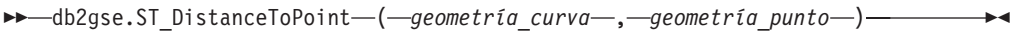
```
SG1_ID SG1_TYPE      SG1_ID SG2_TYPE      DISTANCE
------ ------------- ------ ------------- ------------
     1 ST_Point         101 ST_Point        12373.2168
     1 ST_Point         102 ST_Point        16311.3816
     1 ST_Point         103 ST_Point         9809.4713
     1 ST_Point         110 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, función

La función 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



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

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) seleccionado(s).
```

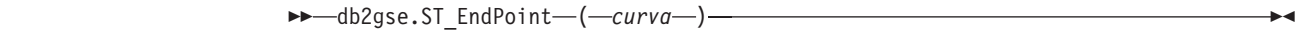
Función ST_Endpoint

La función 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



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)
```

Función ST_Envelope

La función 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.

También se puede invocar esta función como método.

Sintaxis

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))
```

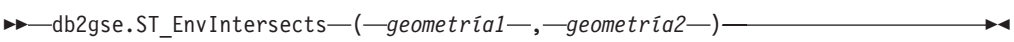
Función ST_EnvIntersects

La función 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



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
```

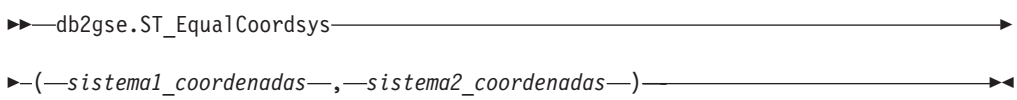
Función ST_EqualCoordsys

La función 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



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
```

Función ST_Equals

La función 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

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:

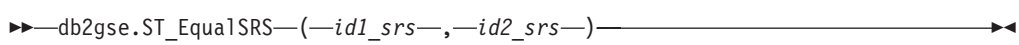
```
ID          ID          EQUALS
----------- ----------- -----------
          5           6           1
```

Función ST_EqualSRS

La función 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



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
```

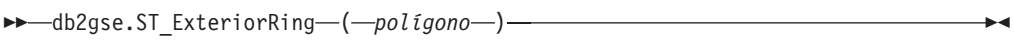
Función ST_ExteriorRing

La función 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



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)
```

Función ST_FindMeasure o ST_LocateAlong

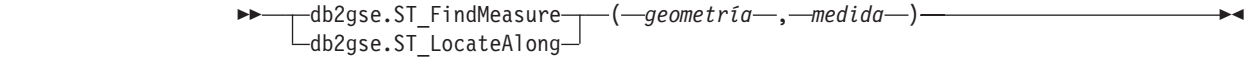
La función ST_FindMeasure o ST_LocateAlong toma una geometría y una medida como parámetros de entrada y devuelve un multipunto o una multicurva que forma parte de la geometría proporcionada y cuya medida es exactamente la especificada.

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 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



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)
```

Función ST_Generalize

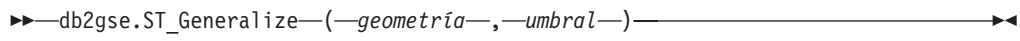
La función ST_Generalize toma una geometría y un umbral como parámetros de entrada y representa dicha geometría con un número reducido de puntos, al mismo tiempo que conserva las características generales de la geometría.

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



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.

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)
```

Función ST_GeomCollection

Utilice la función ST_GeomCollection para construir una colección de geometrías.

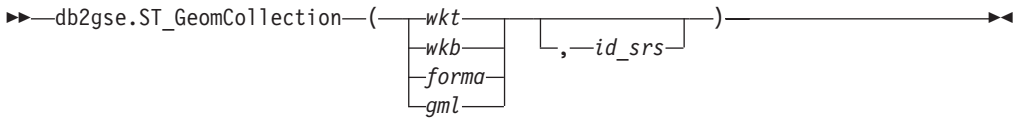
ST_GeomCollection crea una colección de geometrías a partir de uno de estos datos de entrada siguientes:

- 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



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 multilínea 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)
```

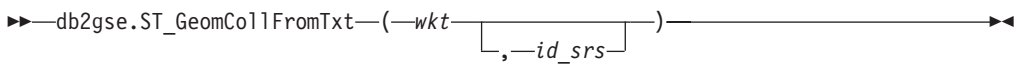
Función ST_GeomCollFromTxt

La función 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



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 multilínea 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)))
```

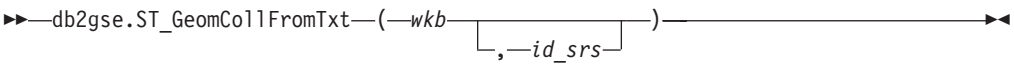
Función ST_GeomCollFromWKB

La función 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



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))
```

Función ST_Geometry

La función ST_Geometry crea una geometría a partir de una representación dada.

ST_Geometry crea una geometría a partir de uno de estos datos de entrada siguientes:

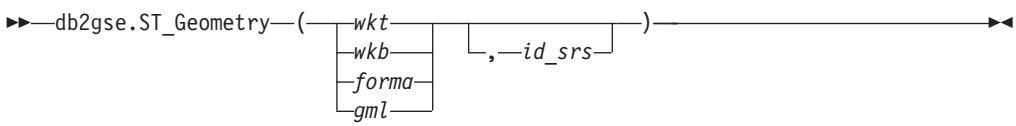
- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma ESRI
- Una representación en formato GML (Geography Markup Language)

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 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 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_WKTToSql 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)
```

Función ST_GeometryN

La función 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

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))
```

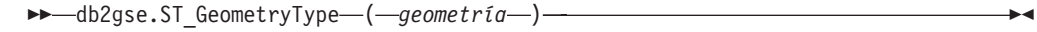
Función ST_GeometryType

La función ST_GeometryType toma una geometría como parámetro de entrada y devuelve el nombre de tipo totalmente calificado 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



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_MULTIPOINT"
```

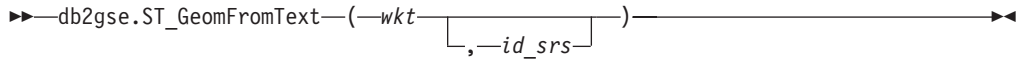
Función ST_GeomFromText

La función 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



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))
```

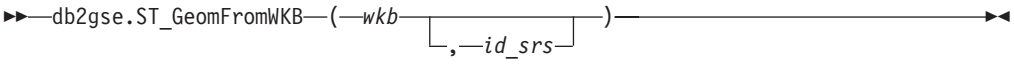
Función ST_GeomFromWKB

La función 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



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))
```

Funciones MBR Aggregate (Agregado MBR)

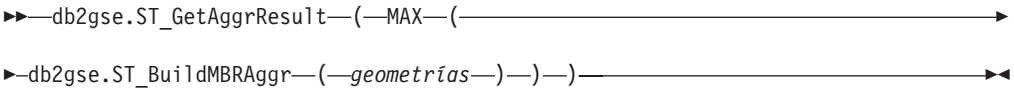
Utilice las funciones ST_BuildMBRAggr y ST_GetAggrResult juntas para agregar un conjunto de geometrías de una columna a una geometría individual. La combinación crea un rectángulo que representa el rectángulo delimitador mínimo (minimum bounding rectangle, MBR) que encierra todas las geometrías contenidas en la columna. Las coordenadas Z y M se descartan cuando se calcula el agregado.

La expresión siguiente es un ejemplo que utiliza la función MAX con la función espacial db2gse.ST_BuildMBRAggr para calcular el rectángulo delimitador mínimo de las geometrías en la columna columnName y la función espacial db2gse.ST_GetAggrResult para devolver la geometría resultante calculada para el rectángulo delimitador mínimo:

```
db2gse.ST_Get_AggrResult(MAX(db2gse.ST_BuildMBRAggr(columnName)))
```

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



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 base de datos particionada
- 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))
```

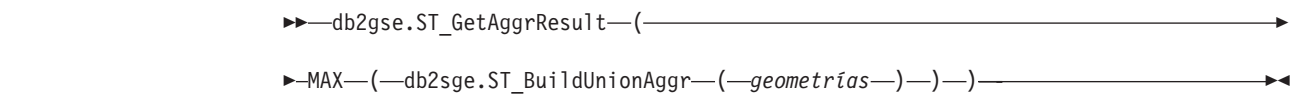
Funciones de agregado de unión

Un agregado de unión es la combinación de las funciones ST_BuildUnionAggr y ST_GetAggrResult. Utilice esta combinación para agregar 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



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 bases de datos particionadas
- 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
```

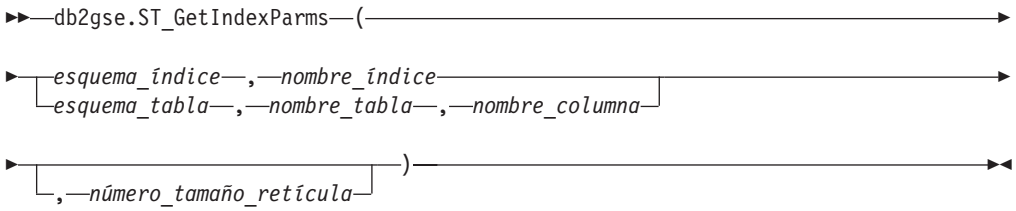
Resultados:

```
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)
```

Función ST_GetIndexParms

La función 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



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
------------------------
  +1.00000000000000E+000
```

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 2)
```

Resultados:

```
1
------------------------
  +1.00000000000000E+001
```

```
VALUES ST_GetIndexParms('SCH', 'IDX', 3)
```

Resultados:

```
1
------------------------
  +1.00000000000000E+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   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'IDX') ) AS t
```

Resultados:

```
ORDINAL     VALUE
----------- ----------------------
          1   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

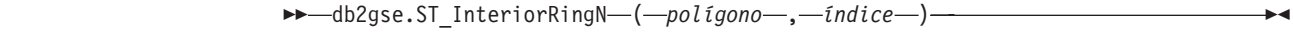
Función ST_InteriorRingN

La función 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



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)
```

Función ST_Intersection

La función 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.

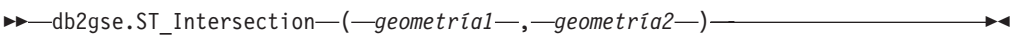
Si es posible, el tipo específico de la geometría devuelta será ST_Point, ST_LineString o ST_Polygon. Por ejemplo, la intersección de un punto y un polígono es vacía o es un punto único, que se representa como ST_MultiPoint.

Si cualquiera de las dos geometrías es un valor nulo, se devuelve un valor nulo.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría y utiliza el sistema de referencia subyacente, se convertirá al otro sistema de referencia espacial.

También se puede invocar esta función como método.

Sintaxis



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*.

Tipo de retorno

db2gse.ST_Geometry

La dimensión de la geometría devuelta es la de la entrada con la dimensión inferior.

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
```

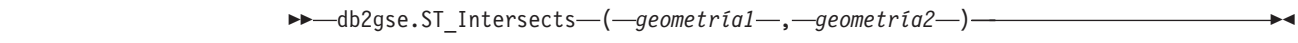
Resultados:

```
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) seleccionado(s).
```

Función ST_Intersects

Utilice la función ST_Intersects para determinar si dos geometrías forman intersección.

Sintaxis



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*.

Tipo de retorno

INTEGER

Uso

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.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría y utiliza el sistema de referencia subyacente, se convertirá al otro sistema de referencia espacial.

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 28. 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 29. 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 30. 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 31. 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	*	*	*

Uso

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_GEOMTRIES1 y SAMPLE_GEOMTRIES2 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
```

Función ST_Is3d

La función 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

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
```

Función ST_IsClosed

La función 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

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
```

Función ST_IsEmpty

La función 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

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
```

Función ST_IsMeasured

La función ST_IsMeasured toma una geometría como parámetro de entrada. Si la geometría dada tiene coordenadas M (medidas), devuelve 1. En caso contrario, devuelve 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

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
```

Función ST_IsRing

La función 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

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
```

Función ST_IsSimple

La función 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

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
```

Función ST_IsValid

La función 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

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
```

Función ST_Length

La función 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

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, la unidad angular asociada a este sistema de coordenadas es el valor por omisión.

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 y se especifica una unidad lineal.

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
```

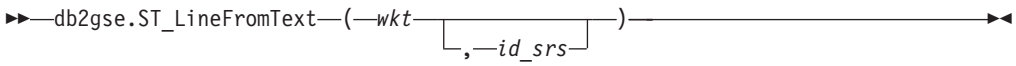
Función ST_LineFromText

La función 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



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
```

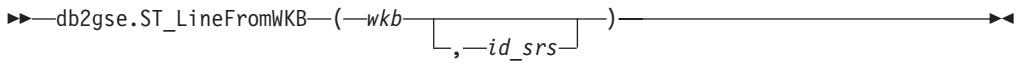
Función ST_LineFromWKB

La función 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



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)
```

Función ST_LineString

La función ST_LineString crea una cadena lineal a partir de una entrada dada.

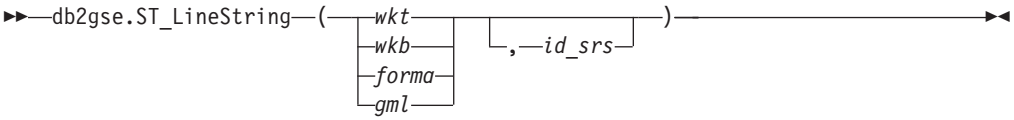
Las entradas se pueden proporcionar con uno de los formatos siguientes:

- 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)
```

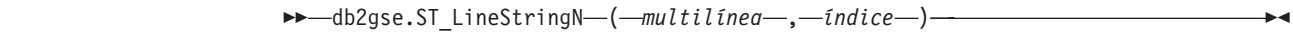
Función ST_LineStringN

La función ST_LineStringN toma una 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



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_LINESTRING
----------- ------------------------------------------------------------------
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)
```

Función ST_M

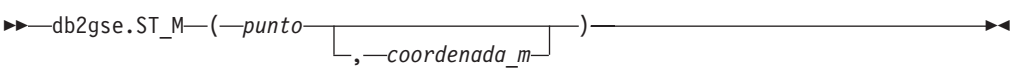
La función ST_M toma un punto como parámetro de entrada y devuelve su coordenada de medida (M). Opcionalmente, puede indicar una coordenada M como parámetro de entrada además del punto, y la función devuelve el propio punto con su coordenada M establecida en el valor especificado.

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



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 +5.00000000000000E+000
          2 +4.00000000000000E+000
          3 +7.00000000000000E+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)
```

Función ST_MaxM

La función 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

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
```

Función ST_MaxX

La función 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_MaxmM(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
```

Función ST_MaxY

La función 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

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
```

Función ST_MaxZ

La función 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

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
```

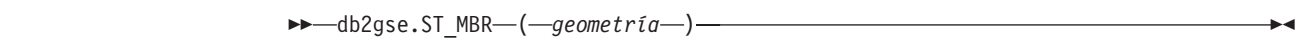
Función ST_MBR

La función 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, se devuelve la propia cadena lineal horizontal o vertical. 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 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 ))
```

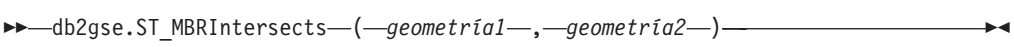
Función ST_MBRIntersects

La función 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



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 MBRs intersect
        1         2 MBRs intersect
        2         2 MBRs intersect
        1         3 MBRs do not intersect
        2         3 MBRs do not intersect
        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
```

Función ST_LocateBetween o ST_MeasureBetween

La función ST_LocateBetween o ST_MeasureBetween 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.

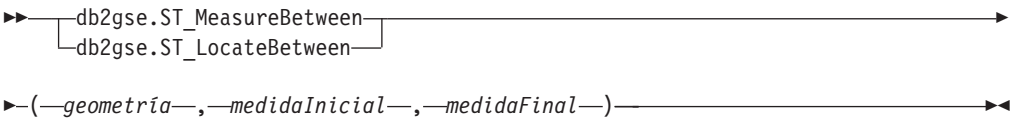
En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

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)
```

Función ST_LocateBetween o ST_MeasureBetween

La función ST_LocateBetween o ST_MeasureBetween 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.

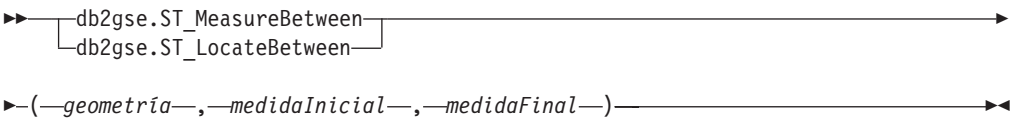
En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

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)
```

Función ST_MidPoint

La función 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. 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

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)
```

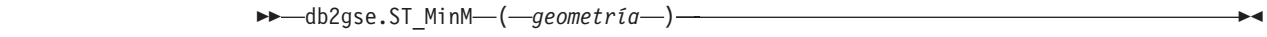
Función ST_MinM

La función 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



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
```

Función ST_MinX

La función 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

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
```

Función ST_MinY

La función 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

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
```

Función ST_MinZ

La función 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

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:

```
OVERALL_MIN_Z
--------------
            10
```

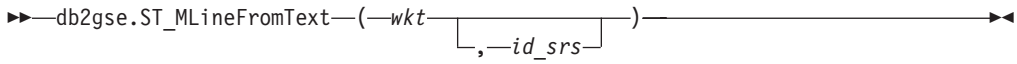
Función ST_MLineFromText

La función 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



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 8.0000000, 43.00000000 12.00000000),
                          ( 39.00000000 3.00000000, 37.00000000 4.00000000,
             36.00000000 7.00000000 ))
```

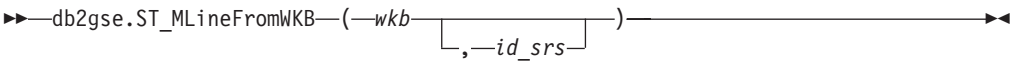
Función ST_MLineFromWKB

La función 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



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 8.0000000),
                           ( 69.00000000 3.00000000, 67.00000000 4.00000000,
              66.00000000 7.00000000, 68.00000000 9.00000000 ))
```

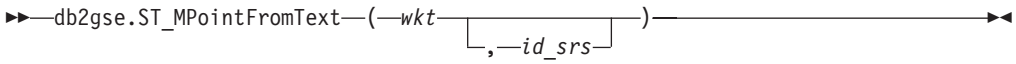
Función ST_MPointFromText

La función 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



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)
```

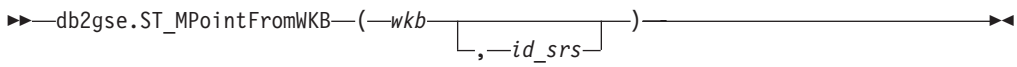
Función ST_MPointFromWKB

La función 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



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)
```

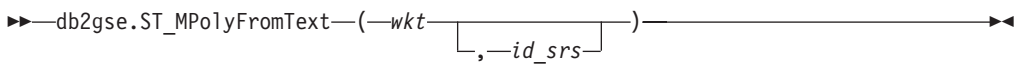
Función ST_MPolyFromText

La función 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



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 28.0000000, 8.00000000 24.00000000)),
           ( 3.00000000 3.00000000, 5.00000000 3.00000000,
4.00000000 6.00000000, 3.00000000 3.00000000)))
```

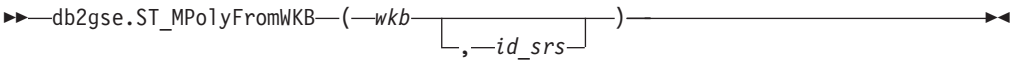
Función ST_MPolyFromWKB

La función 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



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 79.0000000, 1.00000000
              72,00000000)),
                           ( 9.00000000 43.00000000, 6.00000000
               47.00000000, 7.00000000 44.00000000, 9.00000000
               43.00000000 )))
```

Función ST_MultiLineString

La función ST_MultiLineString crea una multilínea a partir de una entrada dada.

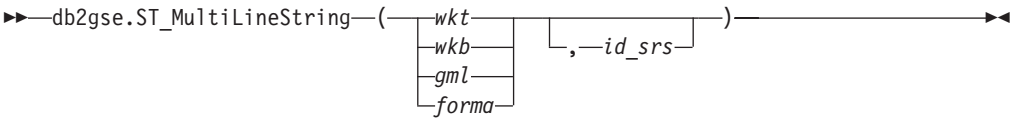
La entrada se puede proporcionar con uno de los formatos siguientes:

- 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 ('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 8.0000000, 43.00000000 12.00000000),
                         ( 39.00000000 3.00000000, 37.00000000 4.00000000,
              36.00000000 7.00000000 ))
```

Función ST_MultiPoint

La función ST_MultiPoint crea un multipunto a partir de una entrada dada.

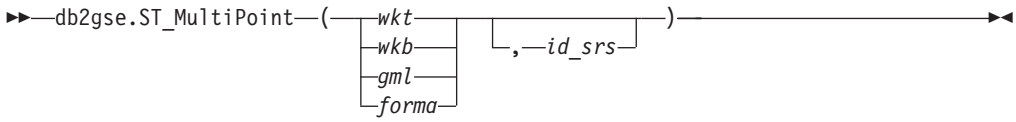
La entrada se puede proporcionar con uno de los formatos siguientes:

- 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)
```

Función ST_MultiPolygon

ST_MultiPolygon construye un multipolígono a partir de una entrada dada.

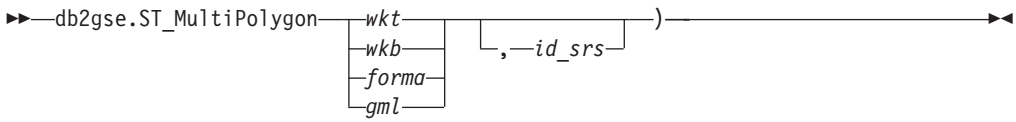
La entrada se puede proporcionar con uno de los formatos siguientes:

- 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 28.0000000, 8.00000000 24.00000000)),
                      (( 3.00000000 3.00000000, 5.00000000 3.00000000,
           4.00000000 6.00000000, 3.00000000 3.00000000)))
```

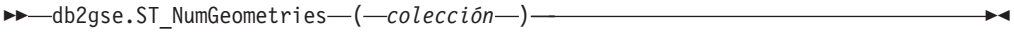
Función ST_NumGeometries

La función 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



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
```

Función ST_NumInteriorRing

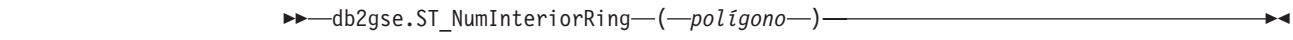
La función ST_NumInteriorRing toma un polígono como parámetro de entrada y devuelve el número de sus anillos interiores.

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



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
```

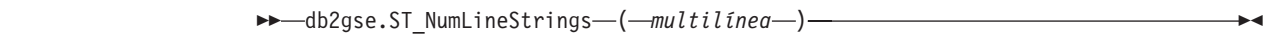
Función ST_NumLineStrings

La función 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



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
```

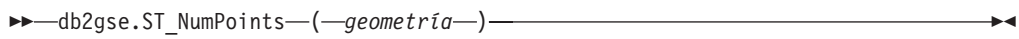
Función ST_NumPoints

La función 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



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
```

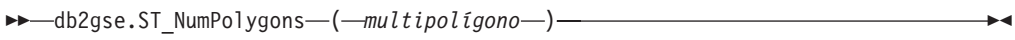
Función ST_NumPolygons

La función 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



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
```

Función ST_Overlaps

La función ST_Overlaps toma dos geometrías como parámetros de entrada. Si la intersección de las geometrías es una geometría de la misma dimensión pero que es diferente de las geometrías originales, devuelve un 1. En caso contrario, devuelve 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

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 compueba 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
```

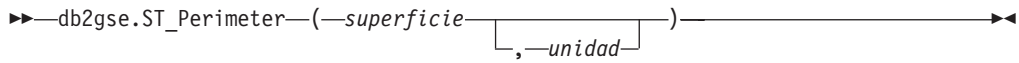
Función ST_Perimeter

La función 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



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, la unidad angular asociada a este sistema de coordenadas es el valor por omisión.

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 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 +1.80000000000000E+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
```

Función ST_PerpPoints

La función 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.

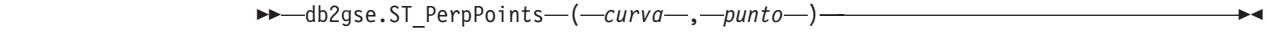
Se obtiene el punto con la distancia más pequeña entre el punto proporcionado y el punto perpendicular. Si dos o más puntos proyectados perpendicularmente son equidistantes respecto al punto proporcionado, se devuelven todos ellos. Si no puede crearse ningún punto perpendicular, se devuelve un punto 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



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 VARCHAR160) ) 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
```

Función ST_Point

La función ST_Point crea un punto a partir de la información de coordenadas o un punto resultante de una representación.

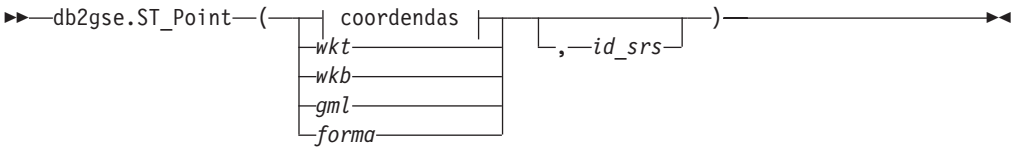
ST_Point obtiene un punto a partir de uno de estos conjuntos de datos de entrada:

- 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)

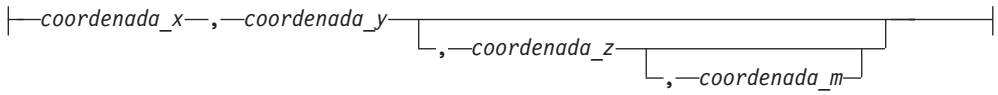
Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenido el punto resultante.

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
---------- -----------------------------------
      1110 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 ( 120.0000000 358.0000000 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:

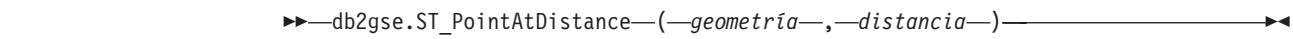
```
ID         POINTS
 ---------- ----------------------------------------------------
       1104 POINT Z ( 1003.000000 9876.000000 20.00000000)
```

ST_PointAtDistance, función

La función 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



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

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) seleccionado(s).
```

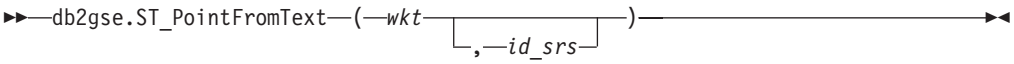
Función ST_PointFromText

La función 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



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)
```

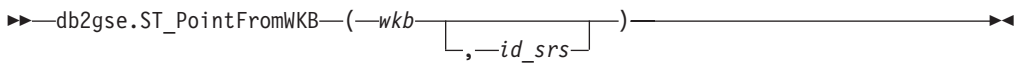
Función ST_PointFromWKB

La función 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



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)
```

Función ST_PointN

La función 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

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)
```

Función ST_PointOnSurface

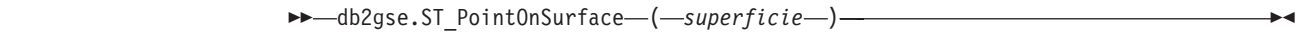
La función ST_PointOnSurface toma como parámetro de entrada una superficie o multisuperficie y devuelve un punto que con toda seguridad estará en el interior de la superficie o multisuperficie. Este punto es el paracentro de la superficie.

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



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)
```

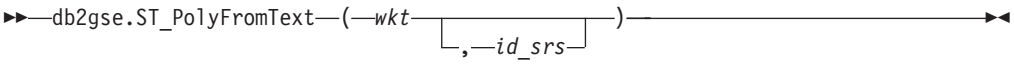
Función ST_PolyFromText

La función 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



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))
```

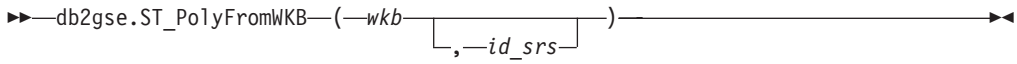
Función ST_PolyFromWKB

La función 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



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))
```

Función ST_Polygon

ST_Polygon construye un polígono a partir de una entrada dada.

La entrada se puede proporcionar con uno de los formatos siguientes:

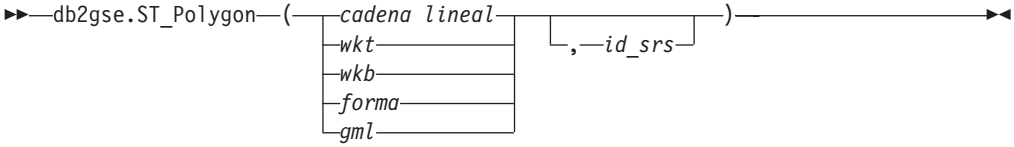
- Una cadena lineal cerrada que define el anillo exterior del polígono resultante
- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

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 38SSL).

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))
```

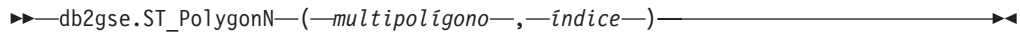
Función ST_PolygonN

La función 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



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))
```

Función ST_Relate

La función ST_Relate toma dos geometrías y una matriz DE-9IM (Dimensionally Extended 9 Intersection Model)como parámetros de entrada. Devuelve 1 si dichas geometrías cumplen las condiciones especificadas por la matriz. De lo contrario, devuelve 0.

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

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**FFF2')  "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
```

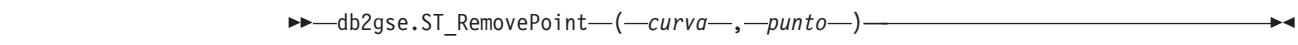
Función ST_RemovePoint

La función 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



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 siguientes.

```
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)
```

Función ST_SrsId o ST_SRID

La función ST_SrsId o ST_SRID toma como parámetros de entrada una geometría y, opcionalmente, un identificador de sistemas de referencia espacial.

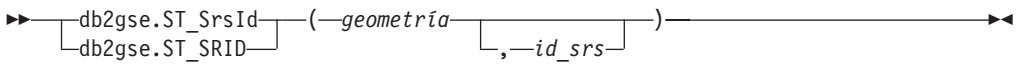
Los datos devueltos dependen de los parámetros de entrada que se hayan especificado:

- 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



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
```

Función ST_SrsId o ST_SRID

La función ST_SrsId o ST_SRID toma como parámetros de entrada una geometría y, opcionalmente, un identificador de sistemas de referencia espacial.

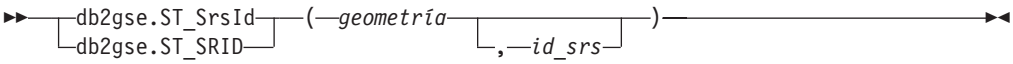
Los datos devueltos dependen de los parámetros de entrada que se hayan especificado:

- 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



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
```

Función ST_SrsName

La función 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

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
```

Función ST_StartPoint

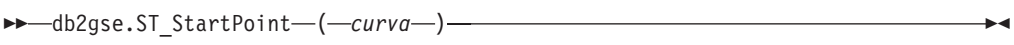
La función ST_StartPoint toma una curva como parámetro de entrada y devuelve el primer punto de la curva.

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



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)
```

Función ST_SymDifference

La función 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.

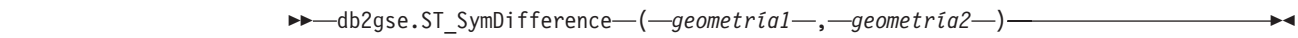
Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría y utiliza el sistema de referencia subyacente, se convertirá al otro sistema de referencia espacial.

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



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*.

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))
```

Función ST_ToGeomColl

La función ST_ToGeomColl toma una geometría como parámetro de entrada y la convierte en un conjunto de geometrías. El conjunto de geometrías resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

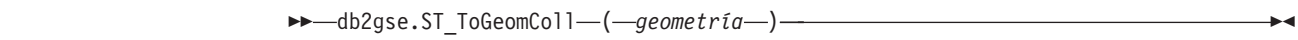
Si la geometría especificada está vacía, puede ser de cualquier tipo. Pero se convierte a ST_Multipoint, ST_MultiLineString o ST_MultiPolygon según convenga.

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



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)
```

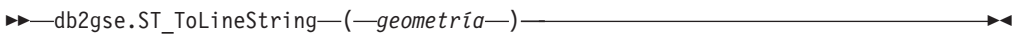
Función ST_ToLineString

La función 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



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
```

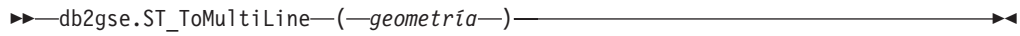
Función ST_ToMultiLine

La función 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



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
```

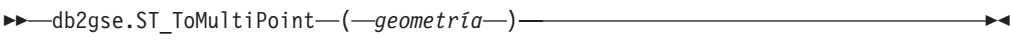
Función ST_ToMultiPoint

La función 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



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
```

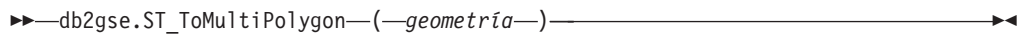
Función ST_ToMultiPolygon

La función 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



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
```

Función ST_ToPoint

La función 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

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
```

Función ST_ToPolygon

La función 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

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
```

Función T_Touches

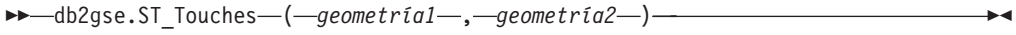
La función ST_Touches toma dos geometrías como parámetros de entrada y devuelve un 1 si las geometrías proporcionadas están espacialmente en contacto. En caso contrario, se devuelve un 0 (cero).

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



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
```

Función ST_Transform

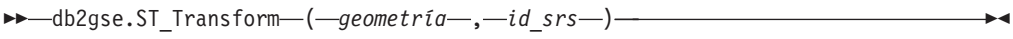
La función ST_Transform toma como parámetros de entrada una geometría y un identificador de sistemas de referencia espacial y transforma la geometría para que esté representada en el sistema de referencia espacial especificado. Se realizan proyecciones y conversiones entre diferentes sistemas de coordenadas 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



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)
```

Función ST_Union

La función 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.

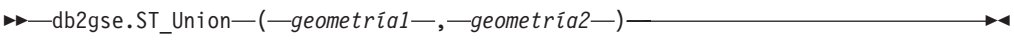
Ambas geometrías deben tener la misma dirección. Si cualquiera de las dos geometrías proporcionadas es nulo, se devuelve un valor nulo.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría y utiliza el sistema de referencia subyacente, se convertirá al otro sistema de referencia espacial.

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



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*.

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
```

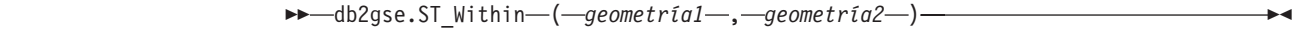
Resultados:

```
ID    ID    UNION
--- --- ------------------------------------------------------------
  4   5 MULTILINESTRING((70.00000000 70.00000000,80.00000000 80.00000000),
           (80.00000000 80.00000000,100.00000000 70.00000000))
```

Función ST_Within

Utilice la función ST_Within para determinar si una geometría está completamente dentro de otra.

Sintaxis



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*.

Tipo de retorno

INTEGER

Uso

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.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría y utiliza el sistema de referencia subyacente, se convertirá al otro sistema de referencia espacial.

ST_Within efectúa la misma operación lógica que ST_Contains, con los parámetros invertidos. ST_Within devuelve el resultado opuesto exacto de ST_Contains.

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 32. Matriz de ST_Within

	Interior Geometría b	**Perím. Geometría b**	**Exterior Geometría b**
Interior Geometría a	V	*	F
Perím. Geometría a	*	*	F
Exterior Geometría a	*	*	*

Ejemplos

La Figura 20 en la página 415 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 20 en la página 415, 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.

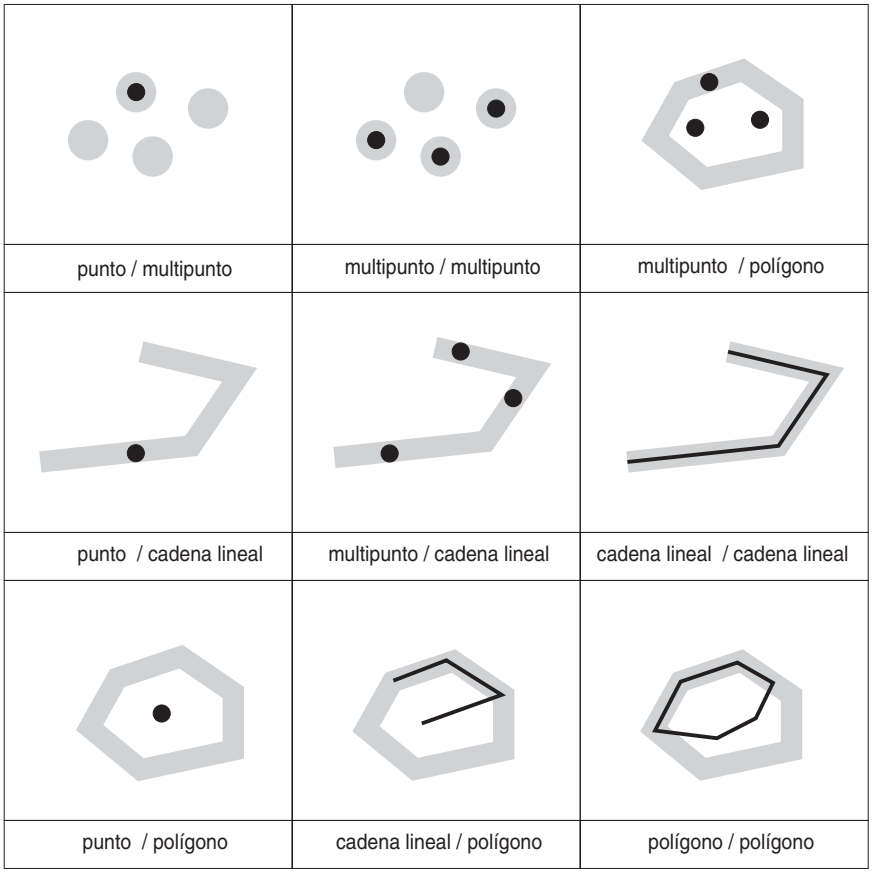


Figura 20. Función ST_Within

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
```

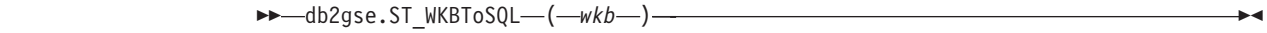
Función ST_WKBToSQL

La función ST_WKBToSQL 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_WKBToSQL(*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_WKBToSQL: ST_Geometry utiliza otras modalidades de datos de entrada además de la representación binaria convencional.

Sintaxis



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_WKBToSQL. 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_WKBToSQL 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))
```

Función ST_WKTToSQL

La función ST_WKTToSQL 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_WKTToSQL(*wkt*) produce el mismo resultado que ST_Geometry(*wkt* ,0). El uso de la función ST_Geometry proporciona más flexibilidad que la función ST_WKTToSQL, ya que ST_Geometry utiliza otras modalidades de datos de entrada además de la representación de texto convencional.

Sintaxis

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_WKTToSQL 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_WKTTSQL ( '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))
```

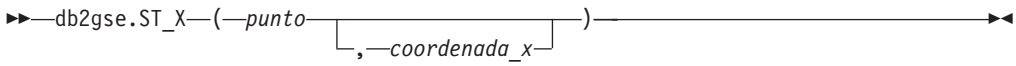
Función ST_X

La función ST_X toma un punto como parámetro de entrada y devuelve su coordenada X. Opcionalmente, puede indicar una coordenada X como parámetro de entrada además del punto, y la función devuelve el propio punto con su coordenada X establecida en el valor especificado.

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



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)
```

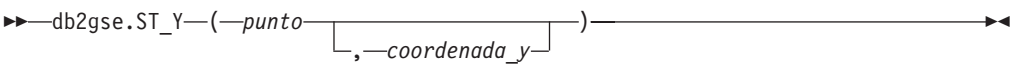
Función ST_Y

La función ST_Y toma un punto como parámetro de entrada y devuelve su coordenada Y. Opcionalmente, puede indicar una coordenada Y como parámetro de entrada además del punto, y la función devuelve el propio punto con su coordenada Y establecida en el valor especificado.

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



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 +3.00000000000000E+000
         2 +5.00000000000000E+000
         3 +8.00000000000000E+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)
```

Función ST_Z

La función ST_Z toma un punto como parámetro de entrada y devuelve su coordenada Y. Opcionalmente, puede indicar una coordenada Z como parámetro de entrada además del punto, y la función devuelve el propio punto con su coordenada Y establecida en el valor especificado.

ST_Z puede actuar de una de estas maneras:

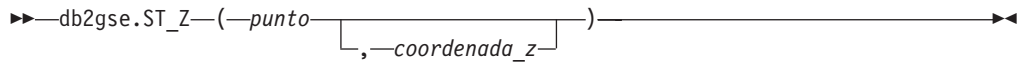
- Toma un punto como parámetro de entrada y devuelve su coordenada Z
- Toma como parámetros de entrada un punto y una coordenada Z y devuelve el propio punto junto con su coordenada Z establecida en el valor proporcionado, aunque el punto especificado no tenga ninguna coordenada Z.

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



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:

```
ID         Z_40
------ ------------------------------------------------------
     3 POINT ZM ( 3.00000000 8.00000000 40.00000000 7.00000000)
```

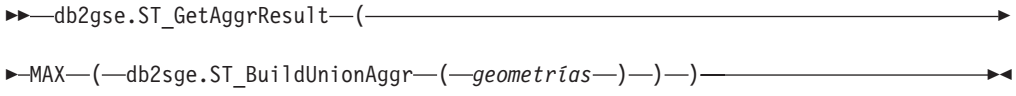
Funciones de agregado de unión

Un agregado de unión es la combinación de las funciones ST_BuildUnionAggr y ST_GetAggrResult. Utilice esta combinación para agregar 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



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 bases de datos particionadas
- 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 19. 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

Utilice el grupo de transformación ST_WellKnownText para transmitir datos hasta y desde DB2 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

© Copyright IBM Corp. 1998, 2012

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 ( 100.00000000 100.00000000, 200.00000000 100.00000000)
```

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'0103000000010000000500000000000000000244000
       0000000000244000000000000002440000000000003440
       0000000000003440000000000000344000000000000034
       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'0100000000000000000034400000000000003E40'
```

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 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 20. Formatos de datos soportados

DB2 Spatial Extender proporciona formatos de datos espaciales industriales aceptados para que pueda utilizarlos.

Se describen estos cuatro formatos de datos espaciales:

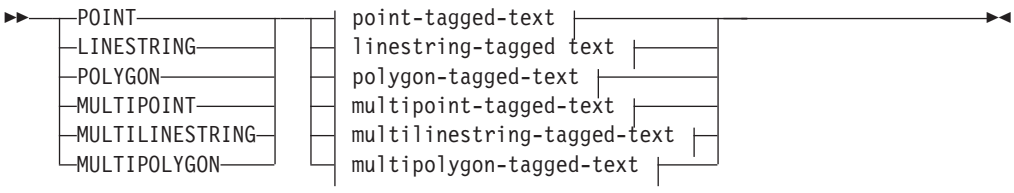
- Representación de texto convencional (well-known text, WKT)
- Representación binaria convencional (well-known binary, WKB)
- Representación de forma
- Representación GML (Geography Markup Language)

Representación de texto convencional (well-known text, WKT)

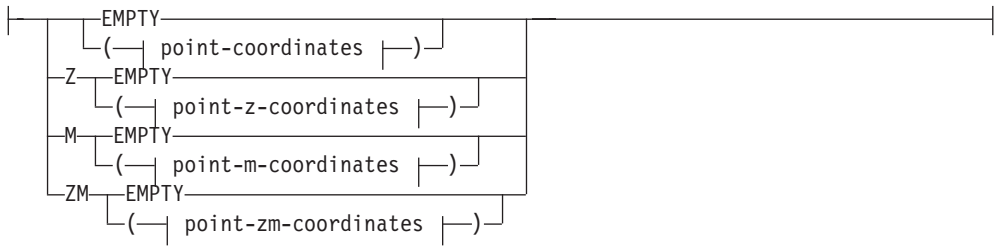
La especificación "Simple Features for SQL" de OpenGIS Consortium define la representación de texto convencional para intercambiar datos de geometría en formato ASCII. Esta representación también está documentada en el estándar "SQL/MM Part: 3 Spatial" de ISO.

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:

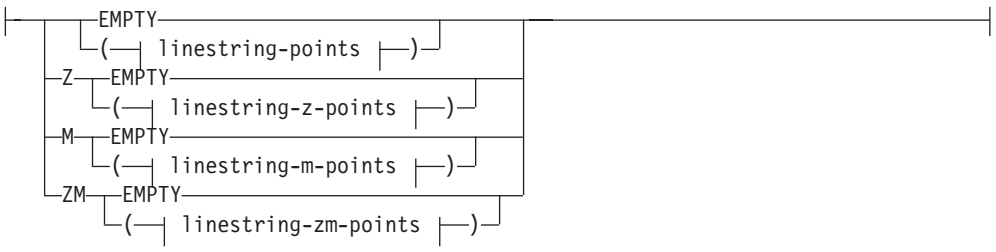


point-tagged-text:

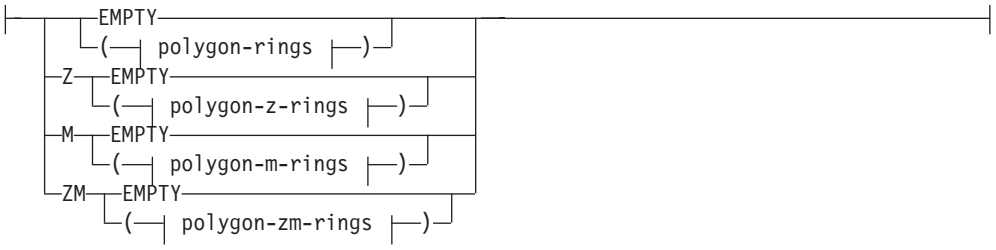


© Copyright IBM Corp. 1998, 2012

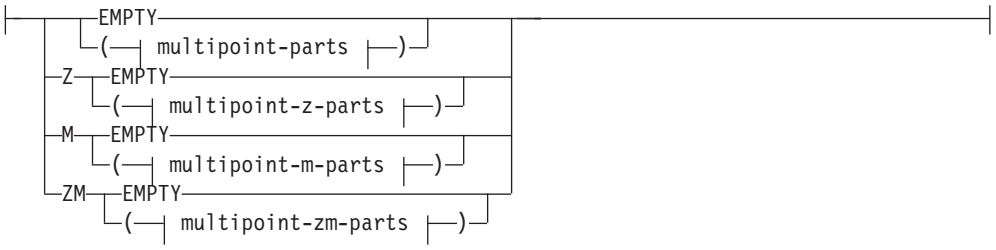
linestring-tagged-text:



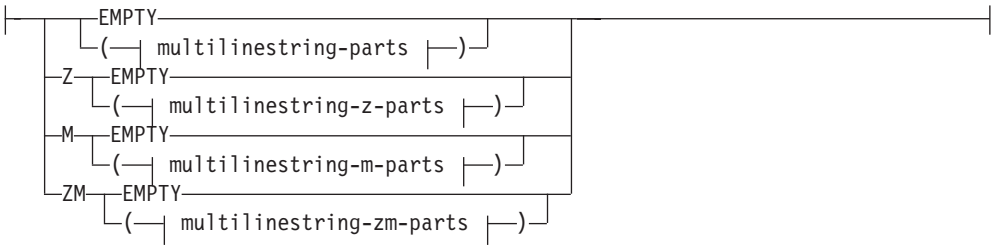
polygon-tagged-text:



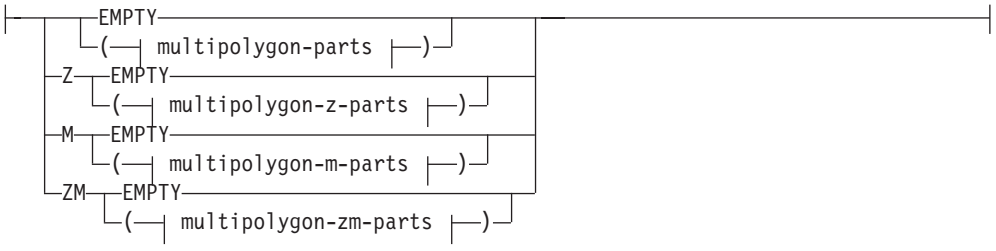
multipoint-tagged-text:



multilinestring-tagged-text:



multipolygon-tagged-text:

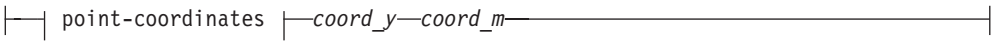


point-coordinates:

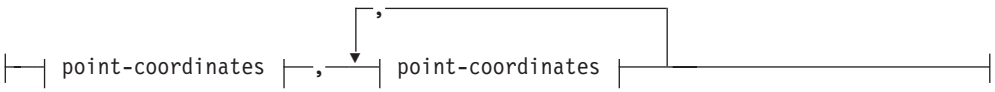
point-z-coordinates:

point-m-coordinates:

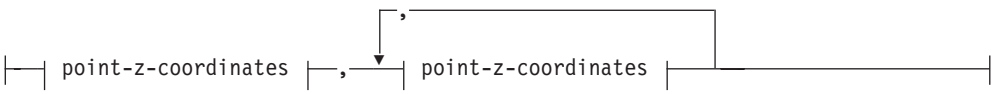
point-zm-coordinates:



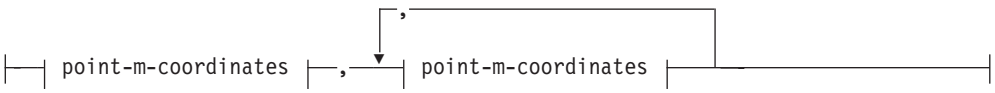
linestring-points:



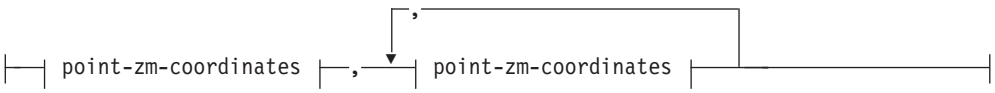
linestring-z-points:



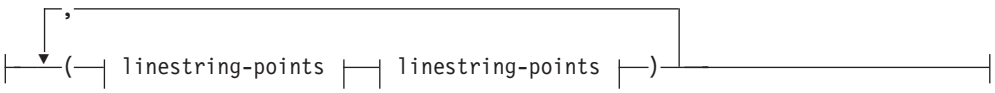
linestring-m-points:



linestring-zm-points:



polygon-rings:



polygon-z-rings:

polygon-m-rings:

polygon-zm-rings:

multipoint-parts:

multipoint-z-parts:

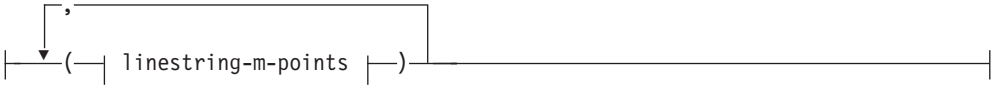
multipoint-m-parts:

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 33. 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 Standarization). 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.

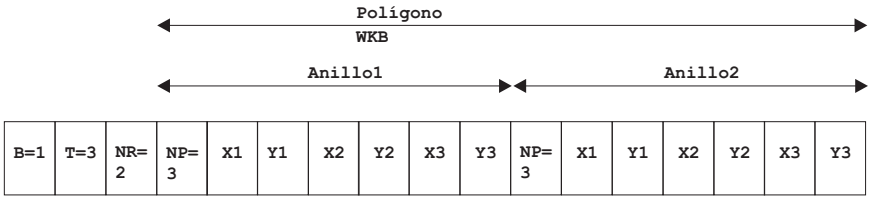


Figura 21. Representación de una geometría en formato NDR. (B=1) de tipo polígono (T=3) con 2 lineales (NR=2), donde cada anillo tiene 3 puntos (NP=3).

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)

GML (Geography Markup Language) es una codificación XML para información geográfica que está definida por la especificación "Geography Markup Language V2" de OpenGIS Consortium.

DB2 Spatial Extender tiene varias funciones que crean geometrías a partir de representaciones en formato GML (Geography Markup Language).

Para obtener más información sobre esta especificación de OpenGIS Consortium, consulte el apartado “OpenGIS Geography Markup Language (GML) Encoding Standard” en http://www.opengeospatial.org/standards/gml.

Capítulo 21. Sistemas de coordenadas soportados

DB2 Spatial Extender utiliza una sintaxis de sistemas de coordenadas específica y valores de sistema de coordenadas soportados para proporcionar una representación textual estándar para la información de los sistemas de coordenadas.

Sintaxis de los sistemas de coordenadas

La sintaxis de los sistemas de coordenadas es una representación en forma de serie de caracteres 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 serie 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

© Copyright IBM Corp. 1998, 2012

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

Utilice unidades lineales que estén soportadas por DB2 Spatial Extender.

Tabla 34. 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

Utilice unidades angulares que estén soportadas por DB2 Spatial Extender.

Tabla 35. 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

Tabla 35. Unidades angulares soportadas (continuación)

Unidad	Rango válido para la latitud	Rango válido para la longitud	Factor de conversión
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

Utilice esferoides que estén soportados por DB2 Spatial Extender.

Tabla 36. 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

Tabla 36. Esferoides soportados (continuación)

Nombre	Eje semimayor	Aplanamiento inverso
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
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

Meridianos de origen soportados

Utilice meridianos de origen que estén soportadas por DB2 Spatial Extender.

Tabla 37. 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

Utilice proyecciones cartográficas que estén soportadas por DB2 Spatial Extender.

Tabla 38. 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 39. 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 (Estereográdica modificada por Snyder)

Tabla 40. 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.
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.

Tabla 40. Parámetros de la proyección cartográfica (continuación)

Parámetro	Descripción
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.

Apéndice A. Visión general de la información técnica de DB2

La información técnica de DB2 está disponible en diversos formatos a los que se puede acceder de varias maneras.

La información técnica de DB2 está disponible a través de las herramientas y los métodos siguientes:

- DB2Centro de información
  - Temas (Tareas, concepto y temas de consulta)
  - 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 la 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 conforme pasen a estar disponibles, o consulte el Centro de información de DB2 en 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. 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. 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.

© Copyright IBM Corp. 1998, 2012

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/e-business/linkweb/publications/servlet/pbi.wss. Los manuales de DB2 Versión 10.1 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.

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 41. Información técnica de DB2

Nombre	Número de documento	Copia impresa disponible	Última actualización
Consulta de las API administrativas	SC11-8067-00	Sí	Abril de 2012
Rutinas y vistas administrativas	SC11-8068-00	No	Abril de 2012
Call Level Interface Guide and Reference Volume 1	SC27-3866-00	Sí	Abril de 2012
Call Level Interface Guide and Reference Volume 2	SC27-3867-00	Sí	Abril de 2012
Consulta de mandatos	SC11-8069-00	Sí	Abril de 2012
Database Administration Concepts and Configuration Reference	SC27-3871-00	Sí	Abril de 2012
Data Movement Utilities Guide and Reference	SC27-3869-00	Sí	Abril de 2012
Database Monitoring Guide and Reference	SC27-3887-00	Sí	Abril de 2012
Data Recovery and High Availability Guide and Reference	SC27-3870-00	Sí	Abril de 2012
Database Security Guide	SC27-3872-00	Sí	Abril de 2012
Guía y consulta de DB2 Workload Management	SC11-8079-00	Sí	Abril de 2012
Developing ADO.NET and OLE DB Applications	SC27-3873-00	Sí	Abril de 2012
Developing Embedded SQL Applications	SC27-3874-00	Sí	Abril de 2012
Desarrollo de aplicaciones Java	SC11-8065-00	Sí	Abril de 2012

Tabla 41. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Última actualización
Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails	SC11-8066-00	No	Abril de 2012
Developing User-defined Routines (SQL and External)	SC27-3877-00	Sí	Abril de 2012
Getting Started with Database Application Development	GI13-2046-00	Sí	Abril de 2012
Iniciación a la instalación y administración de DB2 en Linux y Windows	GI13-1946-00	Sí	Abril de 2012
Globalization Guide	SC27-3878-00	Sí	Abril de 2012
Instalación de servidores DB2	GC11-8073-00	Sí	Abril de 2012
Instalación de clientes de IBM Data Server	GC11-8074-00	No	Abril de 2012
Consulta de mensajes Volumen 1	SC11-8079-00	No	Abril de 2012
Consulta de mensajes Volumen 2	SC11-8080-00	No	Abril de 2012
Net Search Extender Guía de administración y del usuario	SC11-8082-00	No	Abril de 2012
Partitioning and Clustering Guide	SC27-3882-00	Sí	Abril de 2012
pureXML Guide	SC27-3892-00	Sí	Abril de 2012
Spatial Extender Guía del usuario y manual de consulta	SC11-8081-00	No	Abril de 2012
SQL Procedural Languages: Application Enablement and Support	SC27-3896-00	Sí	Abril de 2012
Consulta de SQL - Volumen 1	SC11-8070-00	Sí	Abril de 2012
Consulta de SQL - Volumen 2	SC11-8071-00	Sí	Abril de 2012
Guía de Text Search	SC11-8083-00	Sí	Abril de 2012
Troubleshooting and Tuning Database Performance	SC27-3889-00	Sí	Abril de 2012
Actualización a DB2 Versión 10.1	SC11-8072-00	Sí	Abril de 2012
Novedades en DB2 Versión 10.1	SC11-8078-00	Sí	Abril de 2012
XQuery Reference	SC27-3893-00	No	Abril de 2012

Tabla 42. Información técnica específica de DB2 Connect

Nombre	Número de documento	Copia impresa disponible	Última actualización
DB2 Connect Instalación y configuración de DB2 Connect Personal Edition	SC11-8075-00	Sí	Abril de 2012
DB2 Connect Instalación y configuración de servidores DB2 Connect	SC11-8076-00	Sí	Abril de 2012
Guía del usuario de DB2 Connect	SC11-8077-00	Sí	Abril de 2012

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.

Procedimiento

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

La documentación correspondiente a otras versiones de los productos DB2 se encuentra en otros centros de información en ibm.com.

Acerca de esta tarea

Para los temas de DB2 Versión 10.1, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v10r1.

Para los temas de DB2 Versión 9.8, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/.

Para los temas de DB2 Versión 9.7, 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 9.1, 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 DB2* en el sitio: http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

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 10.1 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:

- Las actualizaciones automáticas actualizan las funciones y los idiomas del Centro de información existentes. Una ventaja de las actualizaciones automáticas es que el Centro de información deja de estar disponible durante un período de tiempo más breve a cuando se realiza la actualización manual. Además, la ejecución de las actualizaciones automáticas se puede configurar como parte de otros trabajos de proceso por lotes que se ejecutan periódicamente.
- Las actualizaciones manuales se pueden utilizar para actualizar las funciones y los idiomas existentes del Centro de información. Las actualizaciones automáticas reducen el tiempo de inactividad durante el proceso de actualización. Sin embargo, debe utilizar el proceso manual cuando desee añadir funciones o idiomas. 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. En el proceso de actualización automática, el Centro de información incurre en una interrupción de servicio para reiniciar el Centro de información solo después de la actualización.

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".

Procedimiento

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/V10.1`.
b. Navegue desde el directorio de instalación al directorio `doc/bin`.
c. Ejecute el script `update-ic`:
   ```
   update-ic
   ```
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\DB2 Information Center\Versión 10.1`, 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 `update-ic.bat`:
      ```
      update-ic.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 localmente, puede obtener e instalar actualizaciones de la documentación de IBM.

Acerca de esta tarea

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 de estación 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**.

Procedimiento

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/db2icdv10 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\DB2 Information Center\Versión 10.1, 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/V10.1.
     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 doc/bin del directorio de instalación 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/db2icdv10 start
     ```

Resultados

El *Centro de información de DB2* actualizado muestra los temas nuevos y actualizados.

Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 le ayudan a conocer diversos aspectos de productos de base de datos 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/db2luw/v10r1/.

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.

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 *Troubleshooting and Tuning Database Performance* o en la sección sobre conceptos fundamentales sobre bases de datos del *Centro de información de DB2*, que contiene:

- Información sobre cómo aislar e identificar problemas con programas de utilidad y herramientas de diagnóstico de DB2.
- Soluciones a algunos de los problemas más comunes.
- Consejo para ayudarle a resolver problemas que podría encontrar en los productos de base de datos DB2

Portal de Soporte de IBM

Consulte el portal de soporte de IBM 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 portal de Soporte de IBM en el sitio http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows

Términos y condiciones

Los permisos para utilizar estas publicaciones se otorgan sujetos a los siguientes términos y condiciones.

Aplicación: Además de las condiciones de uso del sitio web de IBM, se aplican estos 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 derivativos de estas publicaciones, ni reproducirlas, distribuirlas o visualizarlas, ni de partes de las mismas fuera de su empresa, sin el consentimiento expreso de IBM.

Derechos: 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.

Marcas registradas de IBM: IBM, el logotipo de IBM e 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 web en www.ibm.com/legal/copytrade.shtml.

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.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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

© Copyright IBM Corp. 1998, 2012

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
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
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 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. La lista actual de marcas registradas de IBM está disponible en la web, en “Copyright and trademark information”, en la dirección 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 todos los logotipos y marcas registradas basadas en Java son marcas registradas de Oracle, sus filiales o ambos.
- 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, el logotipo de Intel Inside, Celeron, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus empresas subsidiarias en Estados Unidos y en otros países.
- Microsoft, Windows, Windows NT 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

B

C

D

© Copyright IBM Corp. 1998, 2012

E

F

G

H

I

M

N

O

P

R

S

T

U

V

W

SC11-8081-00

Spine information:

IBM DB2 10.1 para Linux, UNIX y Windows

Spatial Extender Guía del usuario y manual de consulta