IBM® DB2® Spatial Extender

Referência e Guia do Usuário

Versão 7

S517-6993-00

IBM® DB2® Spatial Extender

Referência e Guia do Usuário

Versão 7

S517-6993-00

Antes de utilizar estas informações e o produto a que elas se referem, certifique-se de ter lido as informações gerais na seção “Avisos” na página 323.

Índice

Figuras

Tabelas

Sobre este manual

Este manual está dividido em duas partes. A primeira parte contém informações conceituais sobre o DB2 Spatial Extender e explica como instalar, configurar, administrar e programar o DB2 Spatial Extender nos sistemas Windows NT e AIX. A segunda parte consiste em informações de referência sobre procedimentos armazenados, geometrias, funções, mensagens e views de catálogos que você utiliza com o DB2 Spatial Extender.

Quem deveria ler este manual

Este manual é para administradores que configuram o ambiente espacial e para programadores de aplicações que desenvolvem aplicações com os dados espaciais.

Convenções

Este manual utiliza estas convenções para destaque:

Tipo negrito
: Indica comandos e controles da interface gráfica com o usuário (GUI) (por exemplo, nomes de campos, nomes de pastas, opções de menu).

`Tipo monoespaço`
: Indica exemplos de codificação ou de texto digitado por você.

Tipo itálico
: Indica variáveis que devem ser substituídas por um valor. O tipo itálico também indica títulos de manuais e enfatiza palavras.

TIPO MAIÚSCULA
: Indica palavras-chave do SQL e nomes de objetos (como, tabelas, views e servidores).

Como enviar seus comentários

Seu retorno auxilia a IBM a fornecer informações de qualidade. Por favor envie quaisquer comentários que tenha sobre este manual ou qualquer outra documentação DB2. Você pode utilizar qualquer dos seguintes métodos para fornecer comentários:

- Envie seus comentários a partir da Web. Você pode acessar o comentário de leitores online do IBM Data Management em http://www.ibm.com/software/data/rcf

- Envie seus comentários por e-mail para comments@vnet.ibm.com. Assegure-se de que tenha incluído o nome do produto, o número da versão do mesmo e o nome e número da peça (part number), do manual (se aplicável). Caso esteja comentando sobre um texto específico, por favor inclua a localização do texto (por exemplo, um capítulo e um título de seção, um número de tabela, um número de página ou um título de tópico de auxílio).

Parte 1. Utilizando o DB2 Spatial Extender

Capítulo 1. Sobre o DB2 Spatial Extender

Este capítulo apresenta o DB2 Spatial Extender explicando sua finalidade, comentando sobre os dados que ele processa e ilustrando como utilizá-lo. O capítulo termina com um guia rápido do resto do manual.

A finalidade do DB2 Spatial Extender

O DB2 Spatial Extender é utilizado na criação de um *sistema de informações geográficas* (GIS - geographic information system): um complexo de objetos, dados e aplicações que permitem a você gerar e analisar informações espaciais sobre recursos geográficos. Os *recursos geográficos* são os objetos que compõem a superfície da terra e os objetos que a ocupam. Eles compõem o ambiente natural (exemplos são os rios, florestas, montanhas e desertos) e o ambiente cultural (cidades, residências, prédios de escritórios, terrenos e assim por diante).

As *informações espaciais* contêm fatos como:

- A localização de recursos geográficos em relação ao seu meio (por exemplo, pontos dentro de uma cidade onde hospitais e clínicas estão localizados, ou a proximidade das residências da cidade em relação a zonas de terremoto)
- Modos como os recursos geográficos estão relacionados entre si (por exemplo, informações de que um determinado sistema fluvial está contido dentro de um região específica, ou de que determinadas pontes naquela região atravessam os braços do sistema fluvial)
- Medidas que se aplicam a um ou mais recursos geográficos (por exemplo, a distância entre um prédio de escritórios e sua divisão de terreno ou o comprimento de um perímetro de preservação de um pássaro)

As informações espaciais, isoladas ou em conjunto com saídas tradicionais do sistema de gerenciamento de banco de dados relacional (RDBMS), podem ajudá-lo a realizar projetos e fazer negócios e tomar decisões políticas. Suponha, por exemplo, que o gerente de um distrito municipal precise verificar quais requerentes e beneficiários realmente moram dentro da área atendida pelo distrito. O DB2 Spatial Extender pode tirar estas informações a partir da localização da área atendida e dos endereços dos requerentes e beneficiários.

Ou suponha que o proprietário de uma cadeia de restaurantes queira fazer negócio em cidades próximas. Para determinar onde abrir novos restaurantes, o proprietário precisa responder a perguntas como: Em que locais destas cidades está concentrada a clientela que geralmente freqüenta meus

restaurantes? Onde ficam as principais estradas? Onde é mais baixa a taxa de crimes? Onde se encontram os restaurantes da concorrência? O DB2 Spatial Extender pode produzir informações espaciais em exibições visuais para responder estas dúvidas e o RDBMS subjacente pode gerar rótulos e textos para explicar estas exibições.

Outros exemplos do uso do DB2 Spatial Extender aparecem neste manual, especialmente no “Capítulo 7. Recuperando e analisando informações espaciais” na página 55, “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59 e “Capítulo 14. Funções espaciais para consultas SQL” na página 163.

Dados que representam recursos geográficos

Esta seção apresenta uma visão geral dos dados que são gerados, armazenados e manipulados para obtenção de informações espaciais. Os tópicos abrangidos são:

- Como os dados representam os recursos geográficos
- A natureza dos dados espaciais
- Formas de produção de dados espaciais

Como os dados representam os recursos geográficos

No DB2 Spatial Extender, um recurso geográfico pode ser representado por uma linha em uma tabela ou view ou por uma parte desta linha. Considere, por exemplo, dois dos recursos geográficos mencionados no “A finalidade do DB2 Spatial Extender” na página 3, prédios de escritórios e residências. Na Figura 1 na página 5, cada linha da tabela BRANCHES representa uma filial de um banco. Como uma variação, cada linha da tabela CUSTOMERS na Figura 1 na página 5, tomada por inteiro, representa um cliente do banco. No entanto, parte de cada linha — especificamente, as células que contêm o endereço de um cliente — podem ser consideradas como representando a residência do cliente.

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	A	A

Figura 1. Linha da tabela que representa um recurso geográfico; linha da tabela cujos dados do endereço representam um recurso geográfico. A linha de dados na tabela BRANCHES representa uma filial de um banco. As células para dados de endereço na tabela CUSTOMERS representam a residência de um cliente. Os nomes e endereços em ambas as tabelas são fictícios.

As tabelas na Figura 1 contêm dados que identificam e descrevem as filiais e clientes do banco. Tais dados são chamados de *dados do atributo.*

Um subconjunto dos dados do atributo — os valores que denotam os endereços das filiais e dos clientes — podem ser convertidos em valores que resultam em informações espaciais. Por exemplo, como mostrado na Figura 1, o endereço de uma filial é 92467 Airzone Blvd., San Jose CA 95141. O endereço de um cliente é 9 Concourt Circle, San Jose CA 95141. O DB2 Spatial Extender pode converter estes endereços em valores que indicam onde estão situadas a filial e a casa do cliente em relação ao seu meio. A Figura 2 mostra as tabelas BRANCHES e CUSTOMERS com novas colunas que são projetadas para conter tais valores.

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141		A	A

Figura 2. Tabelas com colunas espaciais incluídas. Em cada tabela, a coluna LOCATION irá conter as coordenadas que correspondem aos endereços.

Quando endereços e identificadores similares são utilizados como ponto de partida para informações espaciais, eles são chamados de *dados fonte.* Como os valores derivados deles resultam em informações espaciais, estes valores

derivados são chamados de *dados espaciais*. A seção a seguir descreve os dados espaciais e introduz seus tipos de dados associados.

A natureza dos dados espaciais

Muitos dados espaciais são compostos por coordenadas. Uma *coordenada* é um número que denota uma posição relativa a um ponto de referência. As latitudes, por exemplo, são coordenadas que denotam posições relativas ao equador. As longitudes são coordenadas que denotam posições relativas ao meridiano de Greenwich. Assim, a posição do Yellowstone National Park está definida por sua latitude (44,45 graus ao norte do equador) e sua longitude (110,40 graus oeste do meridiano de Greenwich).

As latitudes, longitudes, seus pontos de referência e outros parâmetros associados são chamados coletivamente de um *sistema de coordenadas*. Também existem sistemas de coordenadas baseados em valores que não sejam a latitude e a longitude. Estes sistemas de coordenadas possuem suas próprias medidas de posição, pontos de referência e parâmetros de distinção adicionais.

O mais simples item de dados espaciais consiste em duas coordenadas que definem a posição de um único recurso geográfico. (Um *item de dados* é o valor ou valores que ocupam uma célula de uma tabela relacional.) Um item de dados espaciais mais amplo consiste em várias coordenadas que definem um caminho linear como uma rua ou rio pode formar. Um terceiro tipo consiste em coordenadas que definem o perímetro de uma área; por exemplo, a margem de um pedaço de terra ou planície aluvial. Estes e outros tipos de itens de dados espaciais que o DB2 Spatial Extender suporta estão descritos mais completamente no “Capítulo 13. Geometrias e funções espaciais associadas” na página 127.

Cada item de dados espaciais é uma instância de um tipo de dados espacial. O tipo de dados para duas coordenadas que marcam a localização é ST_Point; o tipo de dados para coordenadas que definem caminhos lineares é ST_LineString; e o tipo de dados para coordenadas que definem perímetros é ST_Polygon. Estes tipos, juntamente com os outros tipos de dados para dados espaciais, são tipos estruturados que pertencem a uma única hierarquia. Para obter uma visão geral da hierarquia, consulte “Sobre tipos de dados espaciais” na página 33.

De onde vêm os dados espaciais

Os dados espaciais podem ser:

- Derivados de dados do atributo
- Derivados de outros dados espaciais
- Importados

Utilizando dados do atributo como dados fonte

O DB2 Spatial Extender pode obter dados espaciais a partir de dados do atributo, como endereços (conforme mencionado em “Como os dados representam os recursos geográficos” na página 4). Este processo é chamado de *geocodificação*. Para ver a seqüência envolvida, considere Figura 2 na página 5 como uma imagem “antes” e Figura 3 como uma imagem “depois”. A Figura 2 na página 5 mostra que a tabela BRANCHES e a tabela CUSTOMERS, ambas, possuem uma coluna vazia destinada aos dados espaciais. Suponha que o DB2 Spatial Extender efetue o geocode nos endereços nestas tabelas para obter coordenadas que correspondem aos endereços e coloca as coordenadas em colunas. A Figura 3 ilustra este resultado.

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	953 1527	A	A

Figura 3. Tabelas que contêm dados espaciais derivados de dados fonte. A coluna LOCATION na tabela CUSTOMERS contém coordenadas que um geocoder obteve a partir dos endereços nas colunas ADDRESS, CITY, STATE e ZIP. De forma semelhante, a coluna LOCATION na tabela BRANCHES contém coordenadas que o geocoder obteve a partir do endereço nas colunas ADDRESS, CITY, STATE e ZIP desta tabela. Este exemplo é fictício; coordenadas simuladas, e não verdadeiras, são apresentadas.

O DB2 Spatial Extender utiliza uma função, chamada de *geocoder*, para converter dados do atributo em dados espaciais e para colocar estes dados espaciais em colunas da tabela. Para obter maiores informações sobre os geocoders, consulte “Sobre geocodificação” na página 41.

Utilizando outros dados espaciais como dados fonte

Os dados espaciais podem ser gerados não apenas a partir de dados do atributo, mas também de outros dados espaciais. Suponha, por exemplo, que o banco, cujas filiais estão definidas na tabela BRANCHES, deseja saber quantos clientes estão localizados dentro de cinco milhas de cada filial. Para que o banco possa obter estas informações do banco de dados, ele deve fornecer ao banco de dados a definição da zona que se encontra em um raio de cinco milhas ao redor de cada filial. Uma função do DB2 Spatial Extender, ST_Buffer, pode criar esta definição. Utilizando as coordenadas de cada filial como entrada, o ST_Buffer pode gerar as coordenadas que demarcam os

perímetros das zonas desejadas. A Figura 4 mostra a tabela BRANCHES com informações fornecidas pelo ST_Buffer.

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	SALES_AREA
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094	1002 2001, 1192 3564, 2502 3415, 1915 3394, 1002 2001

Figura 4. Tabela que contém novos dados espaciais derivados de dados espaciais existentes. As coordenadas na coluna SALES_AREA foram obtidas pela função ST_Buffer a partir das coordenadas na coluna LOCATION. Assim como as coordenadas na coluna LOCATION, as na coluna SALES_AREA são simuladas; elas não são verdadeiras.

Além do ST_Buffer, o DB2 Spatial Extender fornece várias outras funções que obtém novos dados espaciais de dados espaciais existentes. Para obter descrições do ST_Buffer e destas outras funções, consulte “Funções que geram novas geometrias a partir de existentes” na página 153.

Importando dados espaciais

Um terceiro modo de se obter dados espaciais é importando-os dos arquivos que estejam em um dos formatos suportados pelo DB2 Spatial Extender. Para obter descrições destes formatos, consulte “Capítulo 16. Formatos de arquivos para dados espaciais” na página 297. Estes arquivos contêm dados que geralmente são empregados em mapas: roteiros do censo, planícies aluviais, deslocamentos de terremotos e outros. Ao utilizar tais dados em conjunto com dados espaciais produzidos por você, você pode aumentar as informações geográficas disponíveis. Se, por exemplo, uma departamento de trabalho público precisa determinar a quais riscos uma comunidade residencial está vulnerável, ele pode usar o ST_Buffer para definir uma zona ao redor da comunidade. Depois, o departamento poderia importar dados sobre planícies aluviais e de deslocamentos de terremotos para saber quais destas áreas problemáticas abrangem a zona.

Como criar e usar um GIS do DB2 Spatial Extender

Um GIS do DB2 Spatial Extender é criado através da configuração do DB2 Spatial Extender e do desenvolvimento de projetos do GIS dentro de ambientes combinados do DB2 Spatial Extender e seu DB2 RDBMS subjacente. Você utiliza o GIS através da implementação destes projetos; isto é, gerando e analisando as informações — espaciais e tradicionais — que eles foram projetados para fornecer. O trabalho inteiro envolve a realização de vários conjuntos de tarefas. Esta seção apresenta as interfaces com as quais você pode executar estas tarefas, fornece uma visão geral das tarefas e apresenta um cenário para ilustrá-las.

Interfaces para o DB2 Spatial Extender e funcionalidade associada

Esta seção inspeciona as interfaces mediante as quais você pode criar um GIS do DB2 Spatial Extender (isto é, configurar recursos para ele, obter dados espaciais e assim por diante) e utilizá-lo (isto é, gerar e analisar informações sobre recursos geográficos).

Você pode criar um GIS do DB2 Spatial Extender:

- Utilizando as janelas e opções de menu do DB2 Spatial Extender do Centro de Controle do DB2. Para obter instruções, consulte:
  - “Capítulo 3. Configurando recursos” na página 23
  - “Capítulo 4. Definindo colunas espaciais, registrando-as como camadas e ativando um geocoder para mantê-las” na página 33
  - “Capítulo 5. Preenchendo colunas espaciais” na página 41
  - “Capítulo 6. Criando índices espaciais” na página 53
- Executando um programa de aplicação que chame procedimentos armazenados do DB2 Spatial Extender. Para obter diretrizes sobre o desenvolvimento de um programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.
- Utilizando o Centro de Controle e um programa de aplicação. Você pode usar, por exemplo, o Centro de Controle para chamar o geocoder padrão. Se, além disso, você quiser usar outro geocoder, você primeiro deve registrá-lo para o DB2 Spatial Extender solicitando o procedimento armazenado db2gse.gse_register_gc em um programa de aplicação. (Para obter informações sobre geocoders não-padrões, consulte “Sobre geocodificação” na página 41. Para obter informações sobre o procedimento armazenado db2gse.gse_register_gc, consulte “db2gse.gse_register_gc” na página 91.)
- Utilizando o Centro de Controle, um programa de aplicação, ou ambos, em conjunto com outras interfaces. Para criar, por exemplo, uma tabela para guardar dados que devem ser gerados por uma função espacial, como um geocoder, você pode usar o Processador de Linha de Comandos ou as interfaces do Centro de Controle.

Você pode usar um GIS do DB2 Spatial Extender:

- Transmitindo informações graficamente com um geobrowser; por exemplo, ArcExplorer, que é oferecido pelo Environmental Systems Research Institute (ESRI)
- Submetendo explicitamente consultas SQL a partir do Centro de Controle do DB2 ou do Processador de Linha de Comandos
- Submetendo consultas SQL a partir de um programa de aplicação

Tarefas que você executa para criar e usar um GIS do DB2 Spatial Extender

Esta seção fornece uma visão geral das tarefas através das quais você cria e utiliza um GIS do DB2 Spatial Extender. As tarefas através das quais você cria o GIS envolvem configurar o DB2 Spatial Extender e desenvolver projetos do GIS. As tarefas através das quais você utiliza o GIS envolvem implementar os projetos. Esta visão geral começa com a configuração do DB2 Spatial Extender e depois passa para o desenvolvimento e implementação de um projeto do GIS. A seção termina indicando como as tarefas descritas na visão geral podem variar na prática.

Configurando o DB2 Spatial Extender

Para configurar o DB2 Spatial Extender:

1. Faça planos e preparações (decida quais projetos do GIS serão desenvolvidos, decida o banco de dados a ser ativado para o DB2 Spatial Extender, selecione pessoal para administrar o DB2 Spatial Extender e desenvolva projetos, e outras coisas).
2. Instale o DB2 Spatial Extender.
3. Coloque recursos no lugar para suportar projetos do GIS; por exemplo:

   Recursos fornecidos pelo DB2 Spatial Extender
   Incluem um catálogo do sistema, tipos de dados espaciais, funções espaciais (incluindo um geocoder padrão) e outros. A tarefa de configuração destes recursos é chamada de *ativação do banco de dados para operações espaciais.*

   Geocoders desenvolvidos por usuários, fornecedores ou ambos.
   O geocoder padrão converte endereços dos Estados Unidos em dados espaciais. Sua organização e outras podem fornecer geocoders que convertam endereços estrangeiros e outros tipos de dados do atributo em dados espaciais.

Para obter instruções sobre a instalação do DB2 Spatial Extender, consulte “Capítulo 2. Instalando o DB2 Spatial Extender” na página 17. Para obter instruções sobre o uso do Centro de Controle para colocar os recursos no lugar, consulte “Capítulo 3. Configurando recursos” na página 23. Para obter diretrizes sobre o uso de um programa da aplicação para este fim, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59. Para obter um cenário que ilustre o trabalho geral de configuração do DB2 Spatial Extender, consulte “Um sistema para integrar dados espaciais e tradicionais” na página 13.

Desenvolvendo e implementando um projeto do GIS

Para desenvolver e implementar um projeto do GIS:

1. Faça planos e preparações (defina metas para o projeto, decida as tabelas e dados necessários, determine o sistema ou sistemas de coordenadas a serem utilizados e outras coisas).
2. Decida o sistema ou sistemas de referência espacial a ser utilizado. Valores de coordenada geralmente incluem inteiros positivos, números negativos e números decimais. O DB2 Spatial Extender, no entanto, deve armazenar todos os valores de coordenada na forma de inteiros positivos. Um *sistema de referência espacial* é um conjunto de parâmetros que define como números negativos e decimais em um sistema de coordenadas específico devem ser convertidos em inteiros positivos, para que o DB2 Spatial Extender possa armazená-los. Depois de decidir o sistema de coordenadas a ser utilizado para uma coluna espacial, é preciso especificar o sistema de referência espacial através do qual a conversão necessária pode ocorrer para aquela coluna. Quando um sistema de referência espacial existente satisfaz seus requisitos, você pode utilizá-lo; do contrário, você pode criar um.
3. Defina uma ou mais colunas para conter os dados espaciais, registre-as para o DB2 Spatial Extender e ative um geocoder para mantê-las automaticamente.

   O registro de uma coluna espacial envolve seu registro no catálogo do DB2 Spatial Extender. A partir do momento em que ela é registrada, ela é chamada de *camada*, pois as informações geradas a partir dela acrescentam um estrato ou camada, na paisagem geográfica virtual que o GIS cria para você. Depois de registrá-la, você pode executar operações espaciais na mesma; por exemplo, você pode preenchê-la e definir um índice espacial na mesma.
4. Preencha as colunas espaciais:
   - Para um projeto que precisa de um geocoder, defina parâmetros para o geocoder. Depois, execute-o de modo que, em uma única operação, ele efetue a geocodificação em todos os dados fonte disponíveis e carregue as coordenadas resultantes para a camada.
   - Para um projeto que requer que os dados espaciais sejam importados, importe-os.
5. Facilite o acesso às colunas espaciais. Isto envolve especificamente a definição de índices que permitem que o DB2 acesse dados espaciais rapidamente e a definição de views que permite que os usuários recuperem dados inter-relacionados de modo eficiente. Depois de definir a view, é necessário registrar suas colunas espaciais como camadas.
6. Gere e analise informações espaciais e informações de negócios relacionadas. Isto envolve consultar colunas espaciais e colunas de atributo relacionadas. Em tais consultas, você pode incluir funções do DB2 Spatial Extender que retornam uma grande variedade de informações; por exemplo, a distância mínima entre dois recursos geográficos ou coordenadas que definem uma área ao redor de um recurso geográfico.

Para obter informações sobre a função, ST_Buffer, que retorna tais coordenadas, consulte “Utilizando outros dados espaciais como dados fonte” na página 7 e “ST_Buffer” na página 194. Para obter exemplos de consultas que utilizam funções espaciais, consulte “Capítulo 7. Recuperando e analisando informações espaciais” na página 55 e “Capítulo 14. Funções espaciais para consultas SQL” na página 163.

Para obter instruções sobre o uso do Centro de Controle para realizar tarefas envolvidas no desenvolvimento de um projeto do GIS, consulte:

- “Capítulo 3. Configurando recursos” na página 23
- “Capítulo 4. Definindo colunas espaciais, registrando-as como camadas e ativando um geocoder para mantê-las” na página 33
- “Capítulo 5. Preenchendo colunas espaciais” na página 41
- “Capítulo 6. Criando índices espaciais” na página 53

Para obter diretrizes sobre o uso do Centro de Controle na implementação de um projeto do GIS, consulte “Capítulo 7. Recuperando e analisando informações espaciais” na página 55.

Para obter diretrizes sobre o uso de um programa da aplicação para desenvolver e implementar um projeto do GIS, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Para ver um cenário ilustrativo do trabalho como um todo, consulte “Uma projeto para estabelecer escritórios e ajustar prêmios” na página 13.

Como os conjuntos de tarefas pode variar

Os conjuntos de tarefas realizadas para criar e utilizar um GIS do DB2 Spatial Extender podem variar em conteúdo e seqüência, dependendo dos seus requisitos e das interfaces utilizadas. Considere, por exemplo, as tarefas de definição de colunas para conter dados espaciais, registro das mesmas como camadas e ativação de um geocoder para mantê-las automaticamente. Com o Centro de Controle, estas tarefas podem ser realizadas juntas, a partir de uma única janela. No entanto, se estiver solicitando procedimentos armazenados a partir de um programa, você pode realizar estas tarefas separadamente, e pode regulá-las a seu critério.

Cenário: Uma companhia de seguros atualiza seu GIS

Esta seção apresenta um cenário ilustrativo dos conjuntos de tarefas descritos na seção anterior.

O ambiente de sistemas de informações da Safe Harbor Real Estate Insurance Company contém um sistema DB2 Universal Database e um sistema separado de gerenciamento de banco de dados do GIS. Até certo ponto, as consultas podem recuperar combinações de dados dos dois sistemas. Por exemplo, uma tabela do DB2 armazena informações sobre rendimentos e uma tabela do GIS

armazena as localizações das filiais da empresa. Assim, fica possível descobrir as localizações dos escritórios que apresentam rendimentos de quantias especificadas. Mas os dados dos dois sistemas não podem ser integrados (por exemplo, os usuários não podem juntar colunas do DB2 com colunas do GIS) e serviços do DB2 como otimização de consultas não estão disponíveis ao GIS. Para superar estas desvantagens, a Safe Harbor adquire o DB2 Spatial Extender e estabelece um novo departamento de desenvolvimento do GIS. As seções a seguir descrevem como o departamento configura o DB2 Spatial Extender e leva adiante seu primeiro projeto.

Um sistema para integrar dados espaciais e tradicionais

Para configurar o DB2 Spatial Extender, o departamento de desenvolvimento do GIS da Safe Harbor procede da seguinte maneira:

1. O departamento prepara a inclusão do DB2 Spatial Extender em seu ambiente do DB2. Por exemplo:
   a. A equipe de gerenciamento do departamento aponta uma equipe de administração espacial para instalar e implementar o DB2 Spatial Extender e uma equipe de análise espacial para gerar e analisar informações espaciais.
   b. Como as decisões empresariais da Safe Harbor são conduzidas principalmente pelas exigências dos clientes, a equipe de gerenciamento decide instalar o DB2 Spatial Extender no banco de dados que contém informações sobre seus clientes. Grande parte das informações é armazenada em uma tabela chamada CUSTOMERS.

      Como uma forma conveniente de se referir ao banco de dados selecionado, os membros do departamento de desenvolvimento do GIS o chamam de *banco de dados do GIS*. No entanto, eles estão cientes de que ele não está reservado apenas para projetos do GIS; aplicações não-espaciais podem continuar utilizando-o, como antes.
2. A equipe de administração espacial instala o DB2 Spatial Extender.
3. Esta mesma equipe configura recursos que os projetos do GIS irão exigir:
   - A equipe utiliza o Centro de Controle para fornecer os recursos que ativam o banco de dados do GIS para operações espaciais. Estes recursos incluem o catálogo do DB2 Spatial Extender, os tipos de dados espaciais, as funções espaciais e outros.
   - Como a Safe Harbor está começando a expandir seus negócios para o Canadá, a equipe de administração espacial começa a solicitar aos fornecedores canadenses geocoders que convertem os endereços canadenses em dados espaciais.

Uma projeto para estabelecer escritórios e ajustar prêmios

Para levar adiante seu primeiro projeto do GIS sob o DB2 Spatial Extender, o departamento de desenvolvimento do GIS procede da seguinte maneira:

1. O departamento prepara o desenvolvimento do projeto; por exemplo:

- A equipe de gerenciamento define metas para o projeto:
  - Determinar onde estabelecer novas filiais.
  - Ajustar prêmios com base na proximidade dos clientes às áreas de risco (áreas com altas taxas de acidentes de tráfego, áreas com altas taxas de criminalidade e zonas de enchentes, terremotos e assim por diante).
- O projeto do GIS se preocupa com os clientes e escritórios nos Estados Unidos. Assim, a equipe de administração espacial decide:
  - Utilizar sistemas de coordenadas que definem com precisão as localizações dos setores nos Estados Unidos em que a Safe Harbor faz negócios.
  - Utilizar o geocoder padrão, pois ele está preparado para efetuar o geocode de endereços nos Estados Unidos.
- A equipe de administração espacial decide quais os dados necessários para satisfazer as metas do projeto e em que tabelas estes dados ficarão contidos.

2. Utilizando o Centro de Controle, a equipe cria dois sistemas de referência espacial. Um determina como as coordenadas que definem as localizações dos escritórios devem ser convertidas em itens de dados que o DB2 Spatial Extender pode armazenar. A outra determina como as coordenadas que definem as localizações das residências dos clientes devem ser convertidas em itens de dados que o DB2 Spatial Extender pode armazenar.
3. Utilizando o Centro de Controle, a equipe de administração espacial define colunas para conter os dados espaciais, as registra como camadas e ativa um geocoder para mantê-las automaticamente:
   - A equipe acrescenta uma coluna LOCATION na tabela CUSTOMERS. A tabela já contém endereços dos clientes. O geocoder padrão os converte em dados espaciais e carrega estes dados para a coluna LOCATION.
   - A equipe cria uma tabela OFFICES para conter os dados que agora estão armazenados no GIS separado. Estes dados incluem os endereços das filiais da Safe Harbor, os dados espaciais originados destes endereços através de um geocoder, e dados espaciais que definem uma zona dentro de um raio de cinco milhas ao redor de cada escritório. Os dados gerados pelo geocoder vão para a coluna LOCATION. Os dados que definem as zonas vão para uma coluna SALES_AREA.
   - A equipe registra como camadas as duas colunas LOCATION e as colunas SALES_AREA.
   - A equipe ativa o geocoder padrão para manter automaticamente as duas colunas LOCATION.
4. A equipe de administração espacial preenche a coluna LOCATION da tabela CUSTOMERS, a tabela OFFICES inteira e uma nova tabela HAZARD_ZONES:

- A equipe utiliza o Centro de Controle para preencher a coluna LOCATION da tabela CUSTOMER:
  a. A equipe instrui o geocoder para que ele insira dados espaciais para um endereço na coluna LOCATION somente sob a seguinte condição: uma correspondência entre o endereço e sua contrapartida nos registros da Repartição de Censo dos Estados Unidos (United States Census Bureau) devem ser de 100 por cento de precisão. (Um arquivo de endereços que são fornecidos pela Repartição de Censo é enviado com o DB2 Spatial Extender. Para que o geocoder possa converter um endereço nos dados fonte em dados espaciais, o geocoder deve tentar corresponder este endereço a uma contraparte no arquivo. Os usuários especificam a porcentagem de precisão da correspondência para que o os dados espaciais sejam colocados na tabela. Esta porcentagem é chamada de *precisão*.)
  b. A equipe executa o geocoder no modo batch, para que a realização do geocode de todos os endereços da tabela sejam feita em uma operação. Para desânimo da equipe, o geocoder rejeita aproximadamente um em cada dez endereços!
  c. A equipe supõe que os rejeitados devem ser endereços novos que não possuem correspondência exata nos registros da Repartição de Censo. Para solucionar o problema, a equipe reduz a precisão para 85.
  d. A equipe executa novamente o geocoder no modo batch. A taxa de rejeição dos endereços cai para um nível aceitável.
- Utilizando um utilitário que é fornecido pelo GIS separado, a equipe carrega os dados do escritório em um arquivo. Depois, a equipe utiliza o Centro de Controle para importar estes dados do arquivo para uma nova tabela OFFICES.
- Utilizando o Centro de Controle, a equipe cria uma tabela HAZARD ZONES, registra suas colunas espaciais como camadas e importa os dados para ela. Os dados se originam de um arquivo adquirido de um fornecedor de mapas.

5. Utilizando o Centro de Controle, a equipe de administração espacial facilita o acesso às novas camadas:
   - A equipe cria índices para elas.
   - A equipe cria uma view que junta colunas das tabelas CUSTOMERS e HAZARD ZONES. Depois, a equipe registra como camadas as colunas espaciais da view.
6. A equipe de análise espacial executa consultas para obter informações que irão ajudá-la a satisfazer os objetivos originais: determinar onde estabelecer as novas filiais, e ajustar prêmios com base na proximidade dos clientes às áreas de risco.

Capítulo 2. Instalando o DB2 Spatial Extender

Este capítulo informa as instruções para instalar o DB2 Spatial Extender. Os seguintes tópicos são discutidos:

Configuração do DB2 Spatial Extender

Um sistema do DB2 Spatial Extender é composto pelo DB2 Universal Database, DB2 Spatial Extender e um geobrowser (por exemplo, o ArcExplorer). Geralmente, um banco de dados habilitado para operações espaciais encontra-se no servidor. As aplicações do cliente são utilizadas para acessar dados espaciais através dos procedimentos armazenados e consultas espaciais do DB2 Spatial Extender. Além disso, você pode exibir dados espaciais com um geobrowser.

A Figura 5 ilustra a arquitetura do DB2 Spatial Extender.

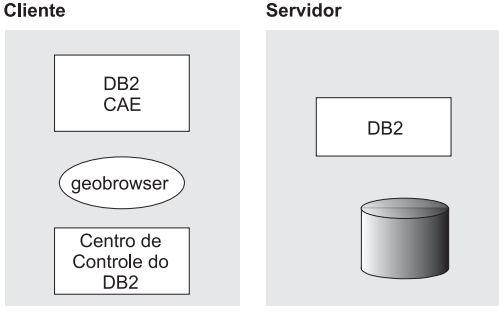


Figura 5. Configuração do cliente-servidor

Exigências do sistema

Esta seção explica a exigências de software e hardware para DB2 Spatial Extender.

Sistemas operacionais suportados

O DB2 Spatial Extender pode ser instalado nos seguintes sistemas operacionais:

- AIX 4.2 ou posterior
- Windows NT 4.0 ou posterior com o Service Pack 5

Software de banco de dados exigido

Antes de instalar o DB2 Spatial Extender, é necessário que o software do DB2 esteja instalado e configurado em seu sistema. A Tabela 1 relaciona as exigências de software do banco de dados para o componente cliente do DB2 Spatial Extender e o componente servidor do DB2 Spatial Extender.

Tabela 1. Exigências mínimas de software

Componente	Software
Cliente	DB2 Administration Client, Versão 7.1[1]
Servidor	Ou: • DB2 Universal Database Enterprise Edition, Versão 7.1 • DB2 Universal Database Enterprise – Extended Edition, Versão 7.1[2]
	Notas: 1. Se *não* estiver pensando em utilizar o Centro de Controle do DB2, um geobrowser para acessar dados espaciais ou o programa de amostra do DB2 Spatial Extender, você pode usar uma versão de nível inferior do DB2 Administration Client. 2. Embora você possa utilizar o DB2 Spatial Extender com o DB2 Universal Database Enterprise - Extended Edition, o índice espacial não pode ser particionado em múltiplos nós como no ambiente de processamento paralelo compacto (MPP).

Exigências de espaço em disco

A Tabela 2 relaciona as exigências de espaço em disco recomendadas para o DB2 Spatial Extender.

Tabela 2. Exigências de espaço em disco

componente do DB2 Spatial Extender	Espaço em disco
Biblioteca do servidor do DB2 Spatial Extender (inclui a biblioteca do servidor do DB2 Spatial Extender, dados de referência do geocoder e documentação)	600 MB
Suporte ao cliente do DB2 Spatial Extender (inclui os dados do programa de amostra)	15 MB

Instalando o DB2 Spatial Extender

Esta seção fornece as informações necessárias para a instalação do DB2 Spatial Extender em sistemas operacionais Windows NT e AIX.

Antes de começar

Caso ainda não o tenha feito, instale o DB2 Administration Client (ferramentas de administração incluindo o Centro de Controle e o cliente de tempo de execução) na estação de trabalho do cliente e instale o DB2 Universal Database Enterprise Edition ou o DB2 Universal Database Enterprise - Extended Edition. Instruções para realizar a instalação encontram-se no manual *Iniciação Rápida* apropriado.

Instalando o DB2 Spatial Extender em sistemas Windows NT

Para instalar o DB2 Spatial Extender em um sistema Windows NT:

1. Efetue logon no sistema com um nome de usuário que possua as permissões de administração necessárias.
2. Encerre todos os outros programas.
3. Insira o CD-ROM na unidade. A barra de lançamento da instalação aparece.
4. Opcional: Clique em **Notas do Release** para verificar nas Notas do Release do DB2 Spatial Extender se há qualquer alteração no processo de instalação, depois retorne à barra de lançamento do DB2 Spatial Extender.
5. Clique em **Instalar**.
6. Responda aos avisos do programa de configuração. O auxílio online está disponível para guiá-lo através dos passos restantes. Para solicitar o auxílio online, clique em **Auxílio** ou pressione a tecla F1.

Quando a instalação estiver completa, o DB2 Spatial Extender é instalado no diretório %DB2PATH% (por exemplo, c:\sqllib).

Instalando o DB2 Spatial Extender em sistemas AIX

Para instalar o DB2 Spatial Extender em um sistema AIX:

1. Efetue login como raiz.
2. Insira o CD-ROM na unidade.
3. Instale o CD-ROM em seu sistema AIX. Para obter informações sobre a instalação de um CD-ROM, consulte o manual *IBM DB2 Universal Database for UNIX Quick Beginnings*.
4. Vá para o diretório no qual o CD-ROM está instalado fornecendo o seguinte comando:

   ```
   cd /cdrom
   ```

   onde *cdrom* é o ponto de instalação da unidade de CD-ROM em AIX.
5. Digite o comando **db2setup** para iniciar o programa instalador do DB2. A janela Instalar DB2 Spatial Extender aparece.

Nota: O programa instalador do DB2 é iniciado lentamente pois ele procura informações em seu sistema.

6. Da lista de produtos na janela Instalar DB2 Spatial Extender, selecione os produtos a serem instalados e clique em **OK**.

   Para obter maiores informações ou assistência durante a instalação do DB2 Spatial Extender, clique em **Auxílio**.

Quando a instalação estiver completa, o DB2 Spatial Extender é instalado no diretório /usr/lpp/db2_07_01.

Verificando a instalação

Depois de instalar o DB2 Spatial Extender, você pode verificar sua instalação utilizando o programa de amostra do DB2 Spatial Extender. Para poder executar o programa de amostra, é necessário criar um banco de dados SAMPLE e tornar o programa de amostra executável.

Nota: Lembre-se de utilizar o compilador especificado no makefile do DB2 Spatial Extender.

Para compilar e executar o programa de amostra para Windows NT:

1. Efetue o login com a ID de usuário que tem privilégios de administrador.
2. Em um prompt da linha de comandos, digite **db2sampl** para criar o banco de dados DB2 SAMPLE.
3. A partir de um prompt de linha de comandos, digite o seguinte comando:

   ```
   cd %DB2PATH%\samples\spatial
   ```

Nota: A fim de realizar a etapa 3 e continuar a verificação da instalação, você deverá possuir a instância de DB2 padrão (DB2–DB2).

4. Digite **make rungsedemo**.
5. Digite **rungsedemo.exe**.
6. Verifique as mensagens de erro e de conclusão que são exibidas à medida que o programa é executado.

Para compilar e executar o programa de amostra para AIX:

1. Efetue login como raiz.
2. Crie ou atualize uma instância do DB2.
3. Em um prompt da linha de comandos, digite **db2sampl** para criar o banco de dados DB2 SAMPLE.
4. A partir de um prompt de linha de comandos, digite o seguinte comando:

   ```
   cd $DB2INSTANCE/sqllib/samples/spatial
   ```

Nota: A fim de realizar a etapa 4 e continuar a verificação da instalação, você deverá possuir a instância do DB2 que criou ou atualizou.

5. Digite **make rungsedemo**.
6. Digite **rungsedemo**.
7. Verifique as mensagens de erro e de conclusão que aparecem à medida que o programa é executado.

Para obter informações sobre os programas de amostra, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Considerações pós-instalação

Depois de instalar o DB2 Spatial Extender com êxito, é preciso pensar em:

- Efetuar o download do ArcExplorer
- Executar o utilitário de atualização de instância do DB2

Efetuando o download do ArcExplorer

A IBM distribui o ArcExplorer Java 3.0 como um programa de amostra ou você pode obtê-lo do Web site do ESRI no endereço http://www.esri.com.

Para obter maiores informações sobre a instalação e uso do ArcExplorer, consulte o manual *Using ArcExplorer*, que também está disponível no Web site do ESRI.

O ArcExplorer requer o Java® 2 Runtime Environment (Standard Edition ou Enterprise Edition), V1.2.2 disponível gratuitamente no Web site da Sun no endereço http://java.sun.com.

Importante: O DB2 Universal Database V7.1 é enviado com o IBM JDK 1.1.8. Ao instalar o JRE 1.2.2 for ArcExplorer, coloque-o em um diretório separado do DB2. Lembre-se de definir corretamente a variável de ambiente CLASSPATH.

Executando o utilitário de atualização de instância do DB2 (db2iupdt)

O utilitário db2iupdt atualiza uma instância especificada do DB2 para:

- Permitir que a instância adquira uma nova configuração do sistema.
- Permitir que a instância acesse uma função associada à instalação ou remoção de determinadas opções do produto.

No AIX, este utilitário está localizado no /usr/lpp/db2_07_01. Se precisar de ajuda, digite `db2iupdt -h` na linha de comandos para abrir o menu de ajuda. No sistema operacional Windows NT, o db2iupdt está localizado no diretório \sqllib\bin. Mude para aquele diretório para digitar o comando. Para obter uma descrição completa deste comando, consulte o manual *IBM DB2 Universal Database Command Reference*.

Qual a próxima etapa?

Depois que o DB2 Spatial Extender estiver instalado, você pode usar o Centro de Controle do DB2 para configurar o ambiente do GIS e começar a trabalhar com informações espaciais.

Para solicitar o DB2 Spatial Extender a partir do Centro de Controle:

1. Da janela do Centro de Controle, expanda a árvore de objetos até encontrar a pasta **Bancos de Dados** sob o servidor no qual deseja que o DB2 Spatial Extender seja executado.
2. Clique na pasta **Bancos de Dados**. Os bancos de dados são exibidos no painel de conteúdo à direita da janela.
3. Dê um clique com o botão direito no banco de dados com o qual deseja trabalhar e, no menu instantâneo, clique na operação espacial que deseja realizar.

Para obter maiores informações sobre o uso do DB2 Spatial Extender a partir do Centro de Controle, consulte:

- "Capítulo 3. Configurando recursos" na página 23
- "Capítulo 4. Definindo colunas espaciais, registrando-as como camadas e ativando um geocoder para mantê-las" na página 33
- "Capítulo 5. Preenchendo colunas espaciais" na página 41
- "Capítulo 6. Criando índices espaciais" na página 53

Capítulo 3. Configurando recursos

Depois de instalar o DB2 Spatial Extender, você está pronto para fornecer ao seu banco de dados os recursos necessários para criar colunas espaciais e manipular dados espaciais. Este capítulo faz uma síntese destes recursos e descreve duas das tarefas através das quais você os torna disponíveis: ativando seu banco de dados para operações espaciais e criando sistemas de referência espacial.

Inventário dos recursos

Os recursos aos quais você recorre para criar colunas espaciais e manipular dados espaciais são:

- *Dados de referência*: endereços que o DB2 Spatial Extender confere para verificar endereços nos quais você deseja efetuar o geocode
- Recursos que ativam um banco de dados para operações espaciais: procedimentos armazenados, funções espaciais e outros
- Geocoders não-padrões fornecidos pelos usuários e fornecedores
- Sistemas de referência espacial

Esta seção discute os dados de referência e os recursos que ativam um banco de dados para operações espaciais. Para obter informações sobre geocoders não-padrões, consulte “Sobre geocodificação” na página 41. Para obter mais informações sobre sistemas de referência espacial, consulte “Sobre sistemas de coordenadas e de referência espacial” na página 25.

Dados de referência

Os *dados de referência* consistem nos endereços mais recentes nos Estados Unidos coletados pela Repartição de Censo do Estados Unidos (United States Census Bureau). Para que o geocoder padrão possa converter em coordenadas um endereço em seu banco de dados, ele deve primeiro igualar parte ou todo o endereço a um endereço nos dados de referência.

Os dados de referência tornam-se disponíveis quando o DB2 Spatial Extender é instalado. Para a quantia de espaço em disco que estes dados requerem, consulte “Exigências de espaço em disco” na página 18. Para verificar no AIX se os dados foram carregados adequadamente, procure-os no diretório $DB2INSTANCE/sqllib/gse/refdata/. Para verificar no Windows NT se os dados foram carregados adequadamente, procure-os no diretório %DB2PATH%\gse\refdata\.

Recursos que ativam um banco de dados para operações espaciais

A primeira etapa que você executa depois de instalar o DB2 Spatial Extender é ativar seu banco de dados para operações espaciais. Isto envolve iniciar uma ação que faz com que o DB2 Spatial Extender carregue o banco de dados com os seguintes recursos:

- Procedimentos armazenados. Quando você solicita uma ação do Centro de Controle, o DB2 Spatial Extender chama um destes procedimentos armazenados para executar a ação.
- Tipos de dados espaciais. Você deve atribuir um tipo de dados espacial a cada coluna da tabela ou view em que os dados espaciais devem ser armazenados. Para obter mais informações, consulte “Sobre tipos de dados espaciais” na página 33.
- Tabelas e views de catálogo do DB2 Spatial Extender. Algumas operações dependem do catálogo do DB2 Spatial Extender. Por exemplo, para que uma coluna com tipo de dados espaciais possa ser preenchida, ela deve ser registrada no catálogo como uma camada. Para obter informações sobre camadas, consulte “Desenvolvendo e implementando um projeto do GIS” na página 10.
- Um tipo de índice espacial. Ele permite que você defina índices para camadas.
- Funções espaciais. Você as utiliza para trabalhar com dados espaciais de diversas formas; por exemplo, para determinar relacionamentos entre recursos geográficos e para gerar mais dados espaciais. Uma destas funções é um geocoder padrão. Ele converte endereços nos Estados Unidos em coordenadas e depois insere estas coordenadas em colunas espaciais. Para obter mais informações sobre funções espaciais, consulte “Capítulo 13. Geometrias e funções espaciais associadas” na página 127 e “Capítulo 14. Funções espaciais para consultas SQL” na página 163. Para obter mais informações sobre o geocoder padrão, consulte “Sobre geocodificação” na página 41.
- Um esquema, chamado DB2GSE, que contém os objetos que acabaram de ser relacionados.

Para obter instruções sobre como usar o Centro de Controle para iniciar o carregamento destes recursos, consulte “Ativando um banco de dados para operações espaciais”. Para obter diretrizes sobre o uso de uma rotina em um programa de aplicação para executar a mesma tarefa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Ativando um banco de dados para operações espaciais

Para descobrir qual autorização é necessária para ativar um banco de dados para operações espaciais, consulte“Autorização” na página 79.

Para ativar um banco de dados para operações espaciais a partir do Centro de Controle:

1. Da janela do Centro de Controle, expanda a árvore de objetos até encontrar a pasta **Bancos de Dados** sob o servidor no qual deseja que o DB2 Spatial Extender seja executado.
2. Clique na pasta **Bancos de Dados**. Os bancos de dados são exibidos no painel de conteúdo à direita da janela.
3. Dê um clique com o botão direito no banco de dados desejado e clique em **Spatial Extender** —> **Ativar** no menu instantâneo. O DB2 Spatial Extender fornece ao banco de dados os recursos que permite a você criar e trabalhar com colunas e dados espaciais.

Lembrete: Para poder ativar um banco de dados para operações espaciais, o DB2 Spatial Extender deve estar instalado no servidor onde o banco de dados reside.

Criando um sistema de referência espacial

Esta seção descreve o relacionamento entre sistemas de referência espacial e sistemas de coordenadas e explica como criar um sistema de referência espacial a partir do Centro de Controle.

Sobre sistemas de coordenadas e de referência espacial

Esta seção continua a discussão sobre sistemas de coordenadas que foi iniciada em “A natureza dos dados espaciais” na página 6. Depois expande-se para definição de sistemas de referência espacial fornecida em “Desenvolvendo e implementando um projeto do GIS” na página 10. Ela também fornece diretrizes para determinação dos valores que devem ser atribuídos aos parâmetros do sistema de referência espacial.

Sistemas de coordenadas, coordenadas e medidas

Você pode imaginar um sistema de coordenadas como sendo uma grade imaginária que encobre uma área geográfica específica. Exemplos seriam uma grade que cobre a terra, uma grade que cobre uma nação ou uma grade que cobre uma região em um estado. Cada recurso geográfico na área está situado na interseção de uma linha da grade leste-oeste e uma norte-sul. Um valor, chamado de *coordenada X*, indica a localização na linha da grade leste-oeste. Outro valor, uma *coordenada Y*, indica a localização na linha da grade norte-sul. Ambos os valores referenciam a localização até o centro da grade ou a *origem*.

As coordenadas X e Y na origem são zero. Da origem rumo leste, as coordenadas X são positivas; da origem rumo oeste, elas são negativas. De modo similar, da origem rumo norte, as coordenadas Y são positivas; da origem rumo sul, elas são negativas. Para uma ilustração desta distribuição, considere o seguinte exemplo generalizado: O sistema de coordenadas A

contém uma grade que abrange uma grande área metropolitana. Uma coordenada X de 7 denotaria uma posição que é de sete unidades de medida rumo leste a partir da origem desta grade. Uma coordenada X de -9.5 denotaria uma posição que é de nove e meio unidades de medida rumo oeste a partir da origem.

Cada item de dados em uma coluna espacial inclui ou (1) uma coordenada X e uma coordenada Y que define a localização de um recurso geográfico ou (2) várias coordenadas X e Y que definem as localizações das partes de um recurso ou que definem a área que um recurso abrange. Dois outros tipos de valores — uma *coordenada Z* e uma *medida* — também podem ser incluídos. Ao contrário das coordenadas X e Y, as coordenadas Z e as medidas não são utilizadas no DB2 Spatial Extender para definir localizações ou áreas. Ao invés disso, elas simplesmente transmitem as informações exigidas por uma aplicação GIS. Uma coordenada Z geralmente indica a altura ou profundidade de um recurso geográfico. As coordenadas Z acima da origem são positivas; as coordenadas Z abaixo dela são negativas. Uma medida é numérica; ela pode transmitir qualquer tipo de informação. Suponha, por exemplo, que você está representando poços de petróleo em seu GIS. Se você solicitar que suas aplicações processem valores que denotam IDs de ponto de detonação para dados sísmicos, você pode armazenar estes valores como medidas.

Sistemas de referência espacial, deslocamentos e fatores de escala

Como indicado no “Sistemas de coordenadas, coordenadas e medidas” na página 25, as coordenadas podem ser negativas e expressas em decimais. O mesmo é verdadeiro para medidas. No entanto, para reduzir armazenamentos suplementares, o DB2 Spatial Extender armazena cada coordenada e medida como um inteiro não-negativo (isto é, como um inteiro positivo ou como zero). Sendo assim, coordenadas e medidas negativas e decimais devem ser convertidas em inteiros não-negativos, para que o DB2 Spatial Extender possa armazená-las. Além disso, é necessário instruir o DB2 Spatial Extender como ele deve fazer a conversão. Isto é feito através da definição de alguns parâmetros. Definições de parâmetro que devem ser utilizadas para converter coordenadas e medidas dentro de uma área geográfica específica são coletivamente chamadas de *sistema de referência espacial*.

Você pode criar um sistema de referência espacial:

- Determinando as coordenadas e medidas negativas mais baixas para os recursos que você está representando. (Quanto mais distante do zero um valor negativo estiver, mais baixo ele é. Uma coordenada X de –10 é mais baixa que uma coordenada X de –5; uma medida de –100 é mais baixa que uma medida de –50.)
- Especificando *fatores de deslocamento* (ou *deslocamentos*, abreviando): valores que, quando subtraídos de coordenadas e medidas negativas, deixam números não-negativos.

- Especificando *fatores de escala*: valores que, quando multiplicados por coordenadas e medidas decimais, resultam em inteiros cuja precisão é pelo menos a mesma que a das coordenadas ou medidas. Considere, por exemplo, uma coordenada com uma precisão de quatro: 92.77. Você poderia multiplicá-la por um fator de escala de 100 para obter um inteiro com uma precisão de quatro: 9277.

Determinando as mais baixas coordenadas e medidas negativas

Antes de definir parâmetros para um sistema de referência espacial, é preciso determinar a mais baixa coordenada X, coordenada Y, coordenada Z e medida negativas na área geográfica que contém os recursos sobre os quais você deseja obter informações. Para descobrir quais são estes valores, responda as seguintes perguntas:

- Dos recursos que você está representando, algum deles fica a oeste da origem do sistema de coordenadas sendo utilizado? Caso haja algum, qual coordenada X indica a localização ou borda oeste do recurso que está mais a oeste? (A resposta será a mais baixa das coordenadas X negativas com as quais você está lidando.) Se, por exemplo, você estivesse representando poços de óleo, e alguns deles ficam a oeste da origem, qual coordenada X indica a localização do poço de petróleo que está na extremidades oeste?
- Há algum recurso a sul da origem? Se houver, qual coordenada Y indica a localização ou borda sul do recurso que está mais a sul? (A resposta será a mais baixa das coordenadas Y negativas com as quais você está lidando.) Se, por exemplo, você está representando poços de óleo, e alguns deles ficam a sul da origem, qual coordenada Y indica a localização do poço de petróleo que está na extremidade sul?
- Se você for utilizar coordenadas Z para definir profundidades, qual recurso é o mais profundo e qual coordenada Z representa o ponto mais baixo deste recurso? (A resposta será a mais baixa das coordenadas Z negativas com as quais você está lidando.)
- Se você for incluir medidas em seus dados espaciais, alguma delas será negativa? Caso seja, qual a mais baixa das medidas negativas?

Tendo determinado as coordenadas e medidas negativas mais baixas, inclua em cada uma uma quantia entre cinco e dez por cento de seu valor. Se, por exemplo, a coordenada X negativa mais baixa for –100, você poderia incluir –5 na mesma. Este manual chama o número resultante de *valor aumentado.*

Especificando valores de deslocamento

Em seguida, especifique quais fatores de deslocamento o DB2 Spatial Extender deve utilizar para converter coordenadas e medidas negativas em não-negativas:

- Depois de decidir qual será o valor X aumentado, especifique um deslocamento que, quando subtraído deste valor, sobre zero. O DB2 Spatial Extender irá então subtrair este número de todas as coordenadas X

negativas até chegar a um valor positivo. O DB2 Spatial Extender também irá subtrair este número de todas as outras coordenadas X.

Se, por exemplo, o valor X aumentado for –105, é preciso subtrair –105 do mesmo para obter 0. Depois, o DB2 Spatial Extender irá subtrair –105 de todas as coordenadas X associadas aos recursos sendo representados. Como nenhuma destas coordenadas é maior que –100, todos os valores que resultam da subtração serão positivos.

- De modo similar, especifique deslocamentos que tenham sobra 0 quando subtraídos do valor Y aumentado, o valor Z aumentado e a medida aumentada.

O deslocamento subtraído das coordenadas X é chamado de *X falso*. Os deslocamentos subtraídos das coordenadas Y, coordenadas Z e medidas são chamados de *Y falso*, *Z falso* e *M falso*, respectivamente. Para obter instruções sobre a especificação destes parâmetros, a partir do Centro de Controle, consulte "Criando um sistema de referência espacial a partir do Centro de Controle" na página 29.

Especificando fatores de escala

Em seguida, especifique os fatores de escala que o DB2 Spatial Extender deve usar para converter coordenadas e medidas decimais em inteiros:

- Especifique um fator de escala que, quando multiplicado por uma coordenada X decimal ou uma coordenada Y decimal, resulte em um inteiro de 32 bits. É aconselhável fazer deste fator de escala um fator de 10: 10 elevado a um (10), 10 elevado a dois (100), 10 elevado a três (1000), ou, se necessário, um fator maior. Para decidir qual fator de 10 o fator de escala deve ser:
  1. Determine quais coordenadas X e Y são, ou provavelmente serão, números decimais. Suponha, por exemplo, que das várias coordenadas X e Y com as quais você estará lidando, você determina que três delas são números decimais: 1.23, 5.1235 e 6.789.
  2. Pegue a coordenada decimal com a precisão decimal mais longa. Depois, determine por qual fator de dez esta coordenada pode ser multiplicada para que resulte em um inteiro de igual precisão. Para ilustrar: das três coordenadas decimais no exemplo atual, 5.1235 possui a precisão decimal mais longa. Multiplicando-a por dez elevado a quatro (10000) o resultado seria o inteiro, 51235.
  3. Determine se o inteiro produzido pela multiplicação que acabou de ser descrita é muito longo para ser armazenado como um item de dados de 32 bits. 51235 não é longo demais. Mas suponha que além de 1.23, 5.11235 e 6.789, a faixa das coordenadas X e Y inclui um quarto valor decimal, 10006.789876. Como esta precisão decimal da coordenada é mais longa do que aquela das outras três, você multiplicaria *esta* coordenada — e não 5.1235 — por um fator de 10. Para convertê-la em um inteiro, você poderia multiplicá-la por 10 elevado a seis (1000000).

Mas o valor resultante, 10006789876, é longo demais para ser armazenado como um item de dados de 32 bits. Se o DB2 Spatial Extender tentasse armazená-lo, os resultados seriam imprevisíveis.

Para evitar este problema, selecione um fator de 10 que, quando multiplicado pela coordenada original, resulte em um número decimal que o DB2 Spatial Extender pode truncar transformando-o em um inteiro armazenável, com perda mínima de precisão. Neste caso, você selecionaria 10 elevado a quatro (10000). A multiplicação de 10000 por 10006.789876 resulta em 100067898.76. O DB2 Spatial Extender truncaria este número transformando-o em 100067898, reduzindo sua precisão por uma quantia virtualmente insignificante.

- Se os recursos que você está representando possuem coordenadas Z decimais, siga o procedimento anterior para determinar um fator de escala para estas coordenadas. Se os recursos estiverem associados a medidas decimais, siga o mesmo procedimento para determinar um fator de escala para estas medidas.

O fator de escala para as coordenadas X e Y é chamado de *unidade XY*. Os fatores de escala para coordenadas Z e medidas são chamados de *unidades Z* e *unidades M*, respectivamente. Para obter instruções sobre a especificação destes parâmetros, a partir do Centro de Controle, consulte “Criando um sistema de referência espacial a partir do Centro de Controle”.

Criando um sistema de referência espacial a partir do Centro de Controle

Esta seção apresenta uma visão geral das etapas para criação de um sistema de referência espacial a partir do Centro de Controle. Após a visão geral seguem detalhes de como completar cada etapa.

Não é necessária nenhuma autorização para realização destas etapas.

Visão geral das etapas para criação de um sistema de referência espacial a partir do Centro de Controle:

1. Abra a janela Criar Referência Espacial.
2. Indique o sistema de coordenadas que você deseja utilizar.
3. Especifique identificadores para o sistema de referência espacial que você deseja criar.
4. Determine que faixas de coordenadas e medidas se aplicam aos recursos geográficos sobre os quais você quer obter informações.
5. Especifique valores que podem ser usados para converter coordenadas e medidas negativas ou decimais em itens de dados que o DB2 Spatial Extender pode armazenar.
6. Peça ao DB2 Spatial Extender para criar o sistema de referência espacial desejado.

Etapas detalhadas para criação do sistema de referência espacial a partir do Centro de Controle:

1. Abra a janela Criar Referência Espacial.
   a. Da janela do Centro de Controle, expanda a árvore de objetos até encontrar a pasta **Bancos de Dados** sob o servidor no qual deseja que o DB2 Spatial Extender seja executado.
   b. Clique na pasta **Bancos de Dados**. Os bancos de dados são exibidos no painel de conteúdo à direita da janela.
   c. Dê um clique com o botão direito no banco de dados que foi ativado para dados espaciais e clique em **Spatial Extender** —> **Referências Espaciais** no menu instantâneo. A janela Referências Espaciais aparece.
   d. A partir da janela Referências Espaciais, clique em **Criar**. A janela Criar Referência Espacial aparece.
2. A partir da janela Criar Referência Espacial, utilize o campo **Sistema de coordenadas** para indicar o sistema de coordenadas que deseja utilizar.
3. Especifique identificadores para o sistema de referência espacial que você deseja criar.
   - No campo **Nome**, digite um nome com 1 a 64 caracteres para o sistema.

     Restrição: Não especifique o nome de outro sistema de referência espacial. Não pode haver dois sistemas de referência espacial no banco de dados com o mesmo nome.
   - No campo da ID, digite um identificador numérico. Ele deve ser um inteiro.

     Restrição: Não especifique a ID de outro sistema de referência espacial. Não pode haver dois sistemas de referência espacial no banco de dados com a mesma ID.
4. Utilizando um meio externo ao Centro de Controle — por exemplo um papel ou um quadro branco — determine as coordenadas e medidas negativas mais baixas que se aplicam aos recursos geográficos sendo representados. Para obter diretrizes sobre como fazer isto, consulte “Determinando as mais baixas coordenadas e medidas negativas” na página 27.
5. A partir da janela Criar Referência Espacial, especifique valores para converter coordenadas e medidas negativas ou decimais em itens e dados que o DB2 Spatial Extender suporte — isto é, em inteiros não-negativos de 32 bits.
   a. Especifique valores para converter coordenadas X negativas ou decimais em inteiros não-negativos:
      - Na coluna **Deslocamento**, no campo mais próximo de **X**, especifique um X falso:

- Se algum dos valores dentro do intervalo de coordenadas X identificados na etapa 4 na página 30 for negativo, digite um X falso que, quando subtraído da mais baixa coordenada negativa, tenha um número positivo como sobra. Para obter orientações, consulte "Especificando valores de deslocamento" na página 27.
- Se todas as coordenadas X forem não-negativas, digite um X falso igual a 0.

• Na coluna **Fator de escala**, especifique uma unidade XY no campo na extremidade direita do **X**. Esta unidade XY deve ser uma que, quando multiplicada por qualquer coordenada X decimal ou coordenada Y decimal, resulta em um número inteiro que pode ser armazenado como um item de dados de 32 bits, com perda mínima de precisão. Para obter orientações, consulte "Especificando fatores de escala" na página 28.

  Depois de especificar a unidade XY no campo na extremidade direita do **X**, ela também aparece no campo na extremidade direita do **Y**.

b. Especifique um Y falso que irá permitir que o DB2 Spatial Extender converta coordenadas Y negativas em valores positivos. Isto é feito na coluna **Deslocamento**, no campo mais próximo do **Y**:
   • Se algum dos valores dentro do intervalo de coordenadas Y identificados na etapa 4 na página 30 for negativo, digite um Y falso que, quando subtraído da coordenada negativa mais baixa, tenha um número positivo como sobra. Para obter orientações, consulte "Especificando valores de deslocamento" na página 27.
   • Se todas as coordenadas Y forem positivas, digite um Y falso igual a 0.
c. Se você for incluir coordenadas Z em seus dados espaciais, especifique valores para converter coordenadas Z negativas ou decimais em inteiros não-negativos:
   • Na coluna **Deslocamento**, no campo mais próximo de **Z**, digite um Z falso:
     – Se algum dos valores dentro do intervalo de coordenadas Z identificados na etapa 4 na página 30 for negativo, digite um Z falso que, quando subtraído da coordenada negativa mais baixa, tenha um número positivo como sobra. Para obter orientações, consulte "Especificando valores de deslocamento" na página 27.
     – Se todas as coordenadas Z forem não-negativas, digite um Z falso igual a 0.
   • Na coluna **Fator de escala**, especifique uma unidade Z no campo na extremidade direita de **Z**. Esta unidade Z deve ser uma que, quando multiplicada por qualquer coordenada Z decimal, resulte em um número inteiro que pode ser armazenado como item de dados de 32

bits, com perda de precisão mínima. Para obter orientações, consulte “Especificando fatores de escala” na página 28.

d. Se você for incluir medidas em seus dados espaciais, especifique valores para converter medidas negativas ou decimais em inteiros positivos:
   - Na coluna **Deslocamento**, no campo mais próximo do rótulo **Linear**, digite um M falso:
     – Se algum dos valores dentro do intervalo de medidas identificados na etapa 4 na página 30 for negativo, digite um M falso que, quando subtraído da medida negativa mais baixa, tenha um número positivo como sobra. Para obter orientações, consulte “Especificando valores de deslocamento” na página 27.
     – Se todas as medidas forem positivas, digite um 0 para M falso.
   - Na coluna **Fator de escala**, especifique uma unidade M no campo na extremidade direita do rótulo **Linear**. Esta unidade M deve ser uma que, quando multiplicada por qualquer medida decimal, resulte em um número inteiro que pode ser armazenado como um item de dados de 32 bits, com perda mínima de precisão. Para obter orientações, consulte “Especificando fatores de escala” na página 28.

6. Clique em **OK** para criar o sistema de referência espacial desejado.

Capítulo 4. Definindo colunas espaciais, registrando-as como camadas e ativando um geocoder para mantê-las

Depois de configurar os recursos para o seu GIS do DB2 Spatial Extender, você está pronto para criar objetos que irão conter dados espaciais. Por exemplo, se precisar de novas tabelas para conter dados espaciais, você pode defini-las, atribuindo tipos de dados espaciais às colunas nas quais deseja que os dados sejam colocados. Se precisar incluir colunas espaciais em tabelas existentes, você também pode fazê-lo.

Quando você fornece uma tabela nova ou existente com uma coluna espacial, é preciso registrar esta coluna como uma camada. Além disso, se pretende ter um geocoder preenchendo a coluna, você pode, no momento em que registra a coluna como uma camada, ativar o geocoder para mantê-la automaticamente.Esta ativação ocorre da seguinte maneira: o DB2 Spatial Extender define disparadores que são codificados para chamar o geocoder sempre que a coluna (ou colunas) do atributo correspondente da coluna espacial recebe dados novos ou atualizados. Quando solicitado, o geocoder converte os dados novos ou atualizados em dados espaciais e coloca estes dados espaciais na coluna espacial.

Depois de definir uma coluna espacial para uma tabela, você pode, se quiser, criar uma coluna de view sobre esta coluna da tabela. Você deve registrar a coluna de view como uma camada depois de registrar a coluna da tabela como uma camada.

Esta capítulo discute a natureza e o uso dos tipos de dados que você pode atribuir à coluna espacial. Em seguida, o capítulo explica como utilizar o Centro de Controle para definir uma coluna espacial para uma tabela, para registrar esta coluna como uma camada e para ativar o geocoder para mantê-la. Por final, o capítulo explica como usar o Centro de Controle para registrar uma coluna de exibição como uma camada.

Sobre tipos de dados espaciais

Esta seção introduz os tipos de dados exigidos para colunas espaciais e fornece diretrizes para escolha de qual deve ser o tipo de dados da coluna espacial.

Quando uma banco de dados é ativado para operações espaciais, o DB2 Spatial Extender fornece ao banco de dados uma hierarquia de tipos de dados estruturados. A Figura 6 na página 34 apresenta esta hierarquia. Nesta figura,

os tipos que podem ser instanciados tem o plano de fundo em branco; os tipos que não podem ser instanciados possuem um plano de fundo sombreado.

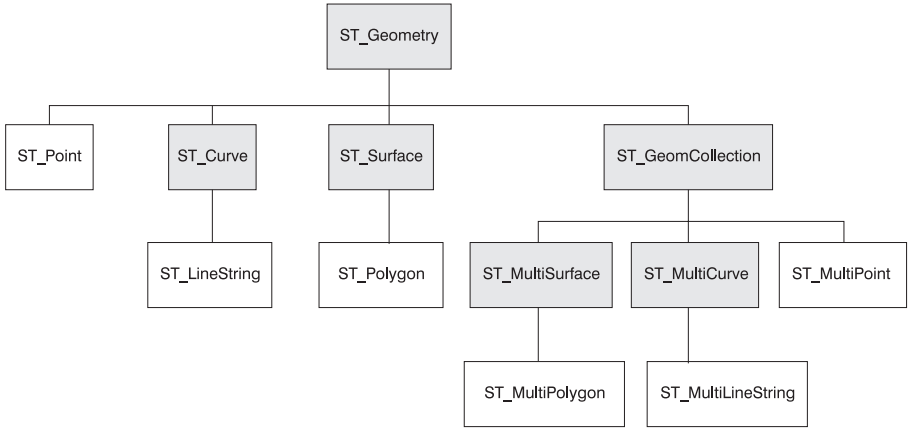


Figura 6. Hierarquia dos tipos de dados espaciais. Tipos de dados em caixas brancas podem ser instanciados. Tipos de dados em caixas sombreadas não podem ser instanciados.

A hierarquia na Figura 6 contém:

- Tipos de dados para recursos geográficos que podem ser captados formando uma única unidade; por exemplo, residências individuais e lagos isolados.
- Tipos de dados para recursos geográficos que são compostos por várias unidades ou componentes; por exemplo, sistemas de estradas e cadeias de montanhas.
- Um tipo de dados para recursos geográficos de todos os tipos.

Tipos de dados para recursos de uma única unidade

Utilize ST_Point, ST_Linestring ST_Polygon para armazenar coordenadas que definem o espaço ocupado pelos recursos que podem ser percebidos como formando uma única unidade:

- Utilize ST_Point quando quiser indicar o ponto no espaço que é ocupado por um recurso geográfico separado. O recurso pode ser muito pequeno, como um poço de água; pode ser muito grande, como uma cidade; ou pode ser intermediário, como um conjunto de prédios ou um parque. Em cada caso, o ponto no espaço pode estar localizado na interseção de uma linha de coordenada leste-oeste (por exemplo, uma paralela) e uma norte-sul (por exemplo, um meridiano). Um item de dados ST_Point contém valores — uma coordenada X e uma coordenada Y — que definem tal interseção. A coordenada X indica onde a interseção está situada na linha leste-oeste; a coordenada Y indica onde a interseção está situada na linha norte-sul.

- Utilize ST_Linestring para coordenadas que definem o espaço que é ocupado por recursos lineares; por exemplo, ruas, canais e canalizações.
- Utilize ST_Polygon quando quiser indicar a extensão do espaço abrangido por um recurso multifacetado; por exemplo, um distrito, uma floresta ou um habitat selvagem. Um item de dados ST_Polygon consiste em coordenadas que definem o perímetro de tal recurso.

  Em alguns casos, ST_Polygon e ST_Point podem ser usados para o mesmo recurso. Por exemplo, suponha que você precise de informações espaciais sobre diversos conjuntos de apartamentos. Se quiser representar o ponto no espaço onde cada complexo está localizado, você usa ST_Point para armazenar as coordenadas X e Y que definem cada ponto. Por outro lado, se quiser representar a área que cada conjunto abrange, você utiliza ST_Polygon para armazenar as coordenadas que definem o perímetro de cada área.

Tipos de dados para recurso de multi-unidades

Utilize ST_MultiPoint, ST_MultiLineString e ST_MultiPolygon para armazenar coordenadas que definem espaços ocupados pelos recursos que são compostos por várias unidades:

- Utilize ST_MultiPoint quando quiser representar recursos compostos por unidades separadas e quiser indicar o ponto no espaço ocupado por cada componente. Um item de dados ST_MultiPoint inclui os pares de coordenadas X e Y que definem a localização de cada componente de tal recurso. Por exemplo, considere uma tabela cujas linhas representam arquipélagos e cujas colunas incluem uma coluna ST_MultiPoint. Cada item de dados nesta coluna contém os pares de coordenadas X e Y que definem as localizações das ilhas em cada arquipélago.
- Utilize ST_MultiLineString quando quiser representar recursos compostos por unidades lineares e quiser obter informações sobre o espaço ocupado por cada unidade. Um item de dados ST_MultiLineString consiste em coordenadas que definem tais espaços. Por exemplo, considere uma tabela cujas linhas representam sistemas de rios e cujas colunas incluem uma coluna ST_MultiLineString. Cada item de dados nesta coluna inclui os conjuntos de coordenadas que definem os caminhos dos rios em cada sistema.
- Utilize ST_MultiPolygon quando quiser representar recursos compostos por unidades multifacetadas e quiser obter informações sobre o espaço ocupado por cada unidade. Por exemplo, considere uma tabela cujas linhas representam municípios do meio-oeste e cujas colunas incluem uma coluna ST_MultiPolygon. Esta coluna contém informações sobre terrenos de fazendas. Especificamente, cada item de dados na coluna contém os conjuntos de coordenadas que definem os perímetros dos terrenos de fazendas em um determinado município.

Um tipo de dados para todos os recursos

Você pode usar ST_Geometry quando não tiver certeza sobre qual dos outros tipos de dados utilizar. Como ST_Geometry é a raiz da hierarquia à qual os outros tipos de dados pertencem, uma coluna ST_Geometry pode armazenar qualquer ou todos os valores que podem ser armazenados em colunas às quais os outros tipos de dados são atribuídos.

Atenção: O geocoder padrão pode converter endereços em itens de dados ST_Point ou ST_Geometry. Sendo assim, se você pretende usar este geocoder para preencher uma coluna espacial, você deve atribuir o tipo de dados ST_Point ou ST_Geometry a esta coluna.

Definindo uma coluna espacial para uma tabela, registrando esta coluna como uma camada e ativando um geocoder para mantê-la

Esta seção apresenta uma visão geral das etapas para definir uma coluna espacial para uma tabela, registrar esta coluna como uma camada e ativar um geocoder para mantê-la. Após a visão geral, seguem detalhes de como completar cada etapa.

Para descobrir de qual autorização você precisa para registrar uma coluna da tabela como uma camada, consulte “Autorização” na página 93. Para descobrir de qual autorização você precisa para ativar um geocoder para manter esta coluna, consulte “Autorização” na página 76.

Visão geral das etapas para definir uma coluna espacial para uma tabela, registrar esta coluna como uma camada e ativar um geocoder para mantê-la:

1. Se a coluna espacial deve fazer parte de uma nova tabela, crie esta tabela.
2. Abra a janela Criar Camada Espacial.
3. Inclua uma coluna espacial em uma tabela e indique que deseja registrar esta coluna como uma camada; ou indique que deseja registrar uma coluna existente como uma camada.
4. Indique qual sistema de referência espacial deve ser usado para a camada.
5. Se a camada deverá conter dados importados ou dados que são gerados de outra coluna espacial, peça ao DB2 Spatial Extender para criar a camada.
6. Se a camada deverá conter dados derivados de dados do atributo:
   a. Especifique qual coluna ou colunas contém estes dados do atributo.
   b. Indique que deseja ativar um geocoder para manter a camada.
   c. Peça ao DB2 Spatial Extender para criar a camada.

Etapas detalhadas para definir uma coluna espacial para uma tabela, registrar esta coluna como uma camada e ativar um geocoder para mantê-la:

1. Se a coluna espacial deve fazer parte de uma nova tabela, crie esta tabela:

- Utilize uma interface de sua escolha (por exemplo, o Centro de Controle ou o Processador de Linha de Comandos) para criar a tabela.
- Se pretende utilizar um geocoder, inclua de uma a dez colunas nas quais o geocoder irá operar. Um geocoder não pode suportar mais do que dez colunas de dados como entrada.
- Inclua a coluna espacial que estará registrando como uma camada ou defina esta coluna na etapa 3.

Se quiser usar uma tabela existente, vá para a etapa seguinte.

2. Abra a janela Criar Camada Espacial.
   a. Da janela Centro de Controle, expanda a árvore de objetos até encontrar a pasta **Tabelas** para as tabelas no banco de dados que você utiliza para operações espaciais.
   b. Clique na pasta **Tabelas**. As tabelas são exibidas no painel de conteúdo à direita da janela.
   c. Dê um clique com o botão direito na tabela desejada e clique em **Spatial Extender** —> **Camadas Espaciais** no menu instantâneo. A janela Camadas Espaciais aparece.
   d. Da janela Camadas Espaciais, clique em **Criar**. A janela Criar Camada Espacial aparece.
3. Da janela Criar Camada Espacial, inclua uma coluna espacial em uma tabela e indique que deseja registrar esta coluna como uma camada; ou indique que deseja registrar uma coluna existente como uma camada.
   - Se quiser incluir uma coluna espacial em uma tabela e definir esta coluna como uma camada:
     a. No campo **Coluna da camada**, digite um nome para a coluna.
     b. No campo **Tipo da coluna**, selecione ou digite o tipo de dados que deseja que a coluna tenha. Para ver uma discussão sobre os tipos de dados permitidos, consulte “Sobre tipos de dados espaciais” na página 33.
   - Se quiser definir uma coluna existente como uma camada, selecione-a no campo **Coluna da camada**.

     Restrição: Não selecione uma coluna que já foi definida como uma camada.
4. No campo **Nome da referência espacial**, especifique o nome do sistema de referência espacial a ser usado para a camada.
5. Caso deseje que a camada contenha dados importados, ou dados que são gerados a partir de outra coluna espacial, clique em **OK** para registrá-la.
6. Caso deseje que a camada contenha dados derivados de dados do atributo:
   a. Especifique qual coluna ou colunas contém estes dados do atributo:

1) Selecione a coluna ou colunas na caixa **Colunas disponíveis**. Você pode selecionar até dez colunas.
2) Clique no botão de comando >, o botão de comando >> ou em ambos para listar a colunas ou colunas selecionadas na caixa **Colunas selecionadas**.

b. Se quiser ativar um geocoder para manter a camada:
1) Selecione a caixa de seleção **Ativar geocoder automático**.
2) No campo **Nome**, selecione o nome do geocoder a ser utilizado.
3) No campo **Nível de precisão**, especifique, em termos de porcentagem, até que grau os registros de entrada devem ser compatíveis aos registros correspondentes nos dados de referência, para serem processados. Esta porcentagem é chamada de *precisão*. Por exemplo, suponha que o geocoder lê um registro de entrada que contém o endereço, 557 Bailey, San Jose 94120. Se a precisão for 100 e se a correspondência entre este endereço e sua contraparte nos dados de referência não for 100 por cento precisa, o geocoder irá rejeitá-la. Se a precisão for 75 e a correspondência entre o registro e sua contraparte de dados de referência for pelo menos 75 por cento precisa, o geocoder irá processá-la.
4) Se o geocoder foi concedido por um fornecedor, utilize a caixa **Propriedades** para especificar todos os parâmetros de geocodificação concedidos pelo fornecedor que você deseja utilizar.

c. Clique em **OK** para registrar a coluna selecionada como uma camada e, se tiver solicitado, para ativar o geocoder para manter a coluna.

Registrando uma coluna de view como uma camada

Para descobrir qual autorização você necessita para registrar uma coluna de view como uma camada, consulte “Autorização” na página 93.

Para registrar uma coluna de view como uma camada:

1. Abra a janela Criar Camada Espacial.
   a. Da janela Centro de Controle, expanda a árvore de objetos até encontrar a pasta **Views** para as views no banco de dados que você utiliza para operações espaciais.
   b. Clique na pasta **Views**. As views são apresentadas no painel de conteúdo à direita da janela.
   c. Dê um clique com o botão direito na view desejada e clique em **Spatial Extender** —> **Camadas Espaciais** no menu instantâneo. A janela Camadas Espaciais aparece.
   d. Da janela Camadas Espaciais, clique em **Criar**. A janela Criar Camada Espacial aparece.

2. Utilize a caixa **Coluna da camada** para especificar a coluna que você deseja registrar como uma camada.
3. No campo **Camada espacial subjacente**, especifique o nome da coluna da tabela na qual está baseada a coluna da view selecionada. Esta coluna da tabela já deve estar registrada como uma camada.
4. Clique em **OK** para registrar a coluna da view especificada como uma camada.

Capítulo 5. Preenchendo colunas espaciais

Após registrar colunas espaciais como camadas, você está pronto para supri-las com dados espaciais. Como observado em “De onde vêm os dados espaciais” na página 6, há três maneiras de fornecer estes dados: use uma função, chamada geocoder, para derivá-lo dos dados de atributo; use outras funções para derivá-lo de outros dados espaciais; ou importe-o de arquivos. Este capítulo:

- Fala sobre geocodificação e explica como usar o Centro de Controle para geocodificar dados de atributo no modo batch.
- Discute a importação e exportação de dados e explica como usar o Centro de Controle para importar dados para o seu GIS e exportá-los a partir do GIS

Para informações sobre as funções que podem derivar novos dados espaciais a partir de dados espaciais existentes, consulte “Funções que geram novas geometrias a partir de existentes” na página 153.

Utilizando os geocoders

Esta seção descreve o processo de geocodificação e explica como executar um geocoder no modo batch a partir do Centro de Controle.

Sobre geocodificação

Esta seção distingue as diferenças básicas entre os geocoders e suas fontes. Ela também descreve os dois modos nos quais um geocoder pode operar e introduz fatores a considerar quando você planeja usar um geocoder.

Com o DB2 Spatial Extender, você pode:

- Usar o geocoder padrão que é fornecido com o DB2 Spatial Extender.
- Ativar geocoders que são desenvolvidos por outros fornecedores.
- Ativar seus próprios geocoders.

O geocoder padrão geocodifica endereços dos Estados Unidos e pode convertê-los em dados ST_Point ou dados ST_Geometry. Se você precisa armazenar dados de outros tipos de dados espaciais, você pode se conectar a um geocoder para gerar tais dados. Se você precisa de dados espaciais que representam locais fora dos Estados Unidos, ou locais que não possuam endereços — por exemplo, área de cultivo que variam por conteúdo de solo — você pode se conectar a um geocoder para encontrar o que precisa.

Antes que um geocoder plug-in possa ser usado, ele deve ser registrado. Usuários e fornecedores podem registrá-lo com o procedimento armazenado

db2gse.gse_register_gc. Ele não pode ser registrado a partir do Centro de Controle. Para obter mais informações sobre db2gse.gse_register_gc, consulte “db2gse.gse_register_gc” na página 91. Para obter informações gerais sobre o uso dos procedimentos armazenados DB2 Spatial Extender, consulte “Capítulo 9. Procedimentos armazenados” na página 69.

Um geocoder opera em dois modos:

- No *modo batch*, ele tenta, em uma operação única, converter todos os dados fontes existentes para uma coluna espacial em dados espaciais e preencher a coluna com aqueles dados. Você pode iniciar esta operação a partir da janela Executar Geocoder. Alternativamente, você pode iniciá-lo em um programa de aplicação, codificando o programa para chamar o procedimento armazenado db2gse.gse_run_gc.
- No *modo incremental*, um geocoder converte dados quando ele é incluído ou atualizado em uma tabela, colocando os valores espaciais resultantes em uma coluna na ordem para manter a atual coluna. Ele é ativado por disparadores de inclusão e atualização que você pode solicitar a partir da janela Criar Camada Espacial. Alternativamente, você pode solicitar o mesmo em um programa de aplicação, codificando o programa para chamar o procedimento armazenado db2gse.gse_enable_autogc.

  A geocodificação incremental é referenciada também como *geocodificação automática*.

Ao planejar o uso de um geocoder, você deve considerar os seguintes fatores:

1. Quando você usa o Centro de Controle, você geralmente usa a janela Criar Camadas Espaciais antes de utilizar a janela Executar o Geocoder. Isto significa que você pode ter definido disparadores DB2 Spatial Extender para a geocodificação incremental antes de iniciar a geocodificação em batch. Portanto, é possível para a geocodificação incremental preceder a geocodificação em batch. Processando todos os dados fontes no modo batch, o geocoder irá geocodificar os mesmos dados que ele operou no modo incremental. Esta redundância não causará duplicações (quando os dados espaciais são produzidos duas vezes, a segunda produção dos dados irá sobrepor a primeira).De qualquer modo, ele pode degradar o desempenho. Uma maneira de evitar isto é postergar a definição dos disparadores até que a geocodificação em batch esteja feita.
2. Se os disparadores estão no lugar quando você estiver pronto para geocodificar no modo batch é aconselhável desativá-los até que a geocodificação em batch tenha encerrado. Você pode desativá-los a partir da janela Executar Geocoder ou em um programa de aplicação, codificando o programa para chamar o procedimento armazenado db2gse.gse_disable_autogc. Se você usa a janela Executar Geocoder, o DB2 Spatial Extender reativa-os automaticamente quando a geocodificação estiver encerrada. Se você usa o procedimento armazenado

db2gse.gse_disable_autogc, você pode reativá-los chamando o procedimento armazenado db2gse.gse_enable_autogc.

3. Se deseja executar um geocoder no modo batch para preencher uma coluna espacial que possui um índice, desative ou elimine o índice primeiro. Caso contrário, se o índice permanece operável enquanto o geocoder é executado, o desempenho será severamente prejudicado. Se estiver usando o Centro de Controle, você pode desativar o índice a partir da janela Executar Geocoder. O DB2 Spatial Extender reativa o índice automaticamente quando a geocodificação estiver encerrada. Se estiver usando um programa de aplicação, você pode eliminar o índice com a instrução SQL DROP. Caso faça isso, assegure-se de tomar nota dos parâmetros do índice, assim você pode recriá-lo após o encerramento da geocodificação.
4. Quando o geocoder lê um registro dos dados fonte, ele tenta fazer coincidir o registro com uma contra-parte nos dados de referência. A coincidência deve ser exata para um certo grau (chamado uma *precisão*) para o geocoder processar o registro. Por exemplo, uma precisão de 85 significa que a coincidência entre um registro fonte e a sua contrapartida nos dados de referência deve ter pelo menos 85 por cento de precisão a fim de que o registro fonte seja processado.

   Você determina qual precisão deve ser. Esteja ciente de que talvez necessite ajustá-lo. Por exemplo, supõem-se que a precisão seja 100. Se muitos registros fonte contêm endereços que são mais recentes do que os dados de referência, a coincidência de 100 por cento entre estes registros e os dados de referência será impossível.Como resultado, o geocoder rejeitará estes registros. No total, se um geocoder produz dados espaciais que parecem insuficientes ou largamente imprecisos, você pode estar apto a resolver este problema alterando a precisão e executando o geocoder novamente.

Executando o geocoder no modo batch

Esta seção oferece uma visão geral das etapas para executar um geocoder no modo batch a partir do Centro de Controle. A visão geral está seguida por detalhes de como completar cada etapa.

Para encontrar a autorização que você precisa para executar um geocoder no modo batch, consulte “Autorização” na página 99.

Visão geral das etapas para executar um geocoder no modo batch:

1. Abra a janela Executar Geocoder.
2. Indique qual geocoder você deseja utilizar.
3. Desative os objetos que possam impedir o desempenho do geocoder.
4. Especifique quantos registros geocodificar antes que o DB2 emita um commit.

5. Indique como deseja que o geocoder opere.
6. instrua o DB2 Spatial Extender a executar o geocoder.

Etapas detalhadas para executar um geocoder no modo batch:

1. Abra a janela Executar Geocoder.
   a. A partir da janela Centro de Controle, expanda a árvore de objetos até encontrar a pasta **Tabelas** em seu banco de dados ativado espacialmente.
   b. Clique na pasta **Tabelas**. As tabelas são exibidas no painel de conteúdos no lado direito da janela.
   c. Clique com o botão direito na tabela que deseja no painel de conteúdo e clique em **Camadas Espaciais** no menu instantâneo. A janela Camadas Espaciais é aberta.
   d. A partir da janela Camadas Espaciais:
      1) Selecione a camada que está definida na coluna que você deseja preencher.
      2) Clique no botão **Executar Geocoder**. A janela Executar Geocoder é aberta.
2. Se você deseja usar o geocoder padrão, deixe a caixa **Nome**, que exibe o nome deste padrão, como está. Caso contrário use a caixa para selecionar o geocoder que deseja.
3. Desative os objetos que possam impedir o desempenho do geocoder:
   - Se a coluna que deseja preencher possui um índice, selecione a caixa de seleção **Desativar temporariamente os índices espaciais durante o processo de geocodificação**.
   - Se os disparadores foram definidos para ativar a geocodificação incremental para esta coluna, selecione a caixa de seleção **Desativar temporariamente os disparadores espaciais durante o processo de geocodificação**.

     O índice e disparadores serão automaticamente reativados ao clicar em **OK** na janela Executar Geocoder.
4. Use o botão de rolagem **Escopo do Commit** para especificar quantos registros geocodificar antes da emissão de um commit do DB2. Por exemplo, se você deseja que o DB2 faça o commit a cada 100 registros, especifique o número 100.

   Dica: Caso você queira o DB2 para emitir somente um commit após todos os registros serem processados, especifique zero.
5. Use os campos na caixa de grupo **Parâmetros do Geocoder** para indicar como deseja que o geocoder opere:
   - Use o botão de rolagem **Nível de Precisão**, para especificar, em termos de porcentagem, quão precisa a coincidência entre registros fonte e suas

contra-partes dos dados de referência devem ser. Para obter maiores informações sobre precisão, consulte “Sobre geocodificação” na página 41.

- Se estiver usando um geocoder fornecido por fornecedor, e deseja usar as propriedades que ela suporta, use a caixa **Propriedades** para definir estas propriedades.
- Caso queira geocodificar apenas um subconjunto de linhas na tabela selecionada, use a caixa **cláusula WHERE** para codificar uma cláusula SELECT WHERE que irá especificar o critério para as linhas que deseja. Esta cláusula pode referenciar quaisquer colunas na tabela.

  Digite somente os critérios. Omita a palavra-chave WHERE. Por exemplo, se a tabela possui uma coluna STATE, e você deseja fazer o geocode apenas para as linhas que contenham o valor MA nesta coluna, digite:

  ```
  STATE='MA'
  ```

6. Clique em **OK** para executar o geocoder.

Importando e exportando dados

Esta seção descreve os processos de importação e exportação de dados, e explica como usar o Centro de Controle para:

- Importar dados de um arquivo de permuta de dados para uma tabela nova ou existente
- Importar dados de um arquivo de permuta de dados para uma tabela existente
- Exportar dados de uma tabela para um arquivo de permuta de dados

Sobre importação e exportação

Esta seção lista as razões para a importação e exportação de dados espaciais. Ele também discute os arquivos de permuta de dados que servem como interfaces entre fontes de exportação e destinos da importação.

Você pode usar o DB2 Spatial Extender para importar dados espaciais de, e exportá-los para, arquivos de permuta de dados. Considere estes cenários:

- Seu GIS contém dados espaciais que representam seus escritórios, clientes e outros assuntos comerciais. Você deseja suplementar estes dados com dados espaciais que representam seu ambiente cultural da organização — cidades, ruas, pontos de interesse, e assim por diante. Os dados que você deseja estão disponíveis a partir de um mapa do fornecedor. Você pode usar o DB2 Spatial Extender para importá-lo de um arquivo de permuta de dados que o fornecedor fornece.
- Você deseja migrar dados espaciais de um sistema Oracle para o seu GIS DB2 Spatial Extender. Você utiliza um utilitário Oracle para carregar os

dados em um arquivo de permuta de dados. Você então usa o DB2 Spatial Extender para importar os dados deste arquivo para o banco de dados que você ativou nas operações espaciais.

- Você deseja usar um navegador GIS para mostrar as apresentações da informação espacial para os clientes. O navegador precisa apenas de arquivos para trabalhar; ele não precisa estar conectado a um banco de dados. Você pode usar o DB2 Spatial Extender para exportar os dados para um arquivo de permuta de dados, e então usar um utilitário de navegação para carregar os dados no navegador.

O Centro de Controle suporta dois tipos de arquivos de permuta de dados para o DB2 Spatial Extender: arquivos de shape e arquivos de transferência ESRI_SDE. Arquivos shape são freqüentemente utilizados para a importação de dados que se originam nos sistemas de arquivo e para a exportação de dados para arquivos que serão carregados nos sistemas de arquivo. Arquivos de transferência ESRI_SDE são usados freqüentemente para a importação de dados que se originam nos bancos de dados ESRI.

Importando dados para uma tabela nova ou existente

Esta seção oferece uma visão geral das etapas para importar dados de um arquivo shape ou de transferência ESRI_SDE para uma tabela nova ou existente. A visão geral está seguida por detalhes de como completar cada etapa.

Para localizar qual autorização é requerida para importar dados shape, consulte “Autorização” na página 89. Para localizar qual autorização é requerida para importar dados ESRI_SDE, consulte “Autorização” na página 87.

Visão Geral das etapas para importar os dados para uma tabela nova ou já existente:

1. Abra a janela Importar Dados Espaciais
2. Especifique o caminho, o nome e o formato do arquivo que contém os dados a serem importados.
3. Especifique quantos registros importar antes de cada commit.
4. Se deseja importar dados espaciais para uma tabela a ser criada, forneça um nome para esta tabela e um nome para a coluna para a qual os dados são planejados. Se a importação dos dados espaciais é para uma tabela existente, indique para qual coluna os dados estão planejados.
5. Especifique qual sistema de referência espacial deve ser associado aos dados.
6. Designe um arquivo para coletar os registros que falhar na importação.
7. Instrua o DB2 Spatial Extender para importar os dados e, se você definiu uma tabela a partir desta janela, para criar a tabela e registrar a coluna para a qual os dados são planejados como uma camada.

Detalhes das etapas para importar os dados para uma tabela nova ou já existente:

1. Abra a janela Importar Dados Espaciais
   a. A partir da janela Centro de Controle, expanda a árvore objeto até localizar a pasta **Bancos de Dados** sob o servidor que você está executando o DB2 Spatial Extender.
   b. Clique na pasta **Bancos de Dados**. Os bancos de dados são exibidos no painel de conteúdos no lado direito da janela.
   c. Clique com o botão direito no banco de dados para o qual deseja importar dados e clique em **Spatial Extender** —> **Importar Dados Espaciais** no menu instantâneo. A janela Importar Dados Espaciais é aberta.
2. Especifique o caminho, o nome e o formato do arquivo que contém os dados a serem importados:
   a. Use o campo **Nome de arquivo** para especificar o caminho e nome.
   b. Use a caixa **Formato de arquivo** para especificar o formato. O formato pode ser:

      Shape Este é o padrão.

      ESRI_SDE
      Se você especificou este formato, o campo **Nome de referência Espacial** usa o padrão para o nome do sistema de referência espacial que está associado a este formato.
3. Use o campo **Escopo do Commit** para especificar o número de registros que você deseja importar antes de cada commit. Por exemplo, para solicitar o commit do DB2 a cada 100 registros, especifique o número 100.

   Dica: Caso você queira o DB2 para emitir somente um commit após todos os registros serem processados, especifique zero.
4. Especifique a tabela e coluna para a qual os dados são planejados.
   a. Use a caixa **Esquema de Camada** para especificar o esquema para a tabela na qual os dados serão importados.
   b. Especifique a tabela e a coluna:
      - Caso a tabela ainda não exista:
        1) No campo **Tabela de Camadas**, digite um nome para a tabela.
        2) No campo **Coluna de Camadas**, digite um nome para a coluna que deverá conter os dados importados. O DB2 Spatial Extender registrará automaticamente esta coluna como uma camada.
      - Caso a tabela já exista:

1) No campo **Tabela de Camadas**, especifique a tabela. Ela já deve conter a coluna para a qual você deseja importar os dados. Em complemento, esta coluna deve já deve estar registrada como uma camada.
2) No campo **Coluna de Camadas**, especifique o nome da coluna para a qual os dados importados está planejado.

5. No campo **Nome de referência espacial**, digite ou selecione o sistema de referência espacial que deve ser associado a estes dados. (Se os dados devem vir do arquivo de transferência ESRI_SDE, o nome do sistema de referência espacial é exibido no campo automaticamente.)
6. No campo **Arquivo de Exceção**, especifique o caminho e nome para um novo arquivo no qual os registros que falharem durante a importação sejam coletados. Mais tarde, você pode alterar estes registros e importá-los deste arquivo.

   O DB2 Spatial Extender criará este arquivo; não especifique um que já exista.
7. Clique em **OK** para importar os dados. Também, se você forneceu um nome para uma tabela que não existe ainda, esta tabela será criada e a coluna para a qual os dados são planejados será registrada como uma camada. Complementarmente, o arquivo de exceção que você especificou será criado.

Importando dados para uma tabela existente

Esta seção oferece uma visão geral das etapas para importar dados de um arquivo shape ou arquivo de transferência ESRI_SDE para uma tabela existente. A visão geral está seguida por detalhes de como completar cada etapa.

Para localizar que autorização é requerida para importar dados shape, consulte “Autorização” na página 89. Para localizar qual autorização é requerida para importar dados ESRI_SDE, consulte “Autorização” na página 87.

Visão Geral das etapas para importar os dados para uma tabela existente:

1. Abra a janela Importar Dados Espaciais
2. Especifique o caminho e o nome do arquivo que contém os dados a serem importados.
3. Especifique quantos registros importar antes de cada commit.
4. Especifique a coluna que conterá os dados espaciais que você está importando.
5. Especifique qual sistema de referência espacial deve ser associado a estes dados.
6. Designe um arquivo para coletar registros que falharem na importação.

7. Instrua ao DB2 Spatial Extender para importar os dados e, se você especificou uma coluna que não tenha sido criada ainda, para criar esta coluna e registrá-la como uma camada.

Detalhes das etapas para importar os dados para uma tabela existente:

1. Abra a janela Importar Dados Espaciais
   a. A partir da janela Centro de Controle, expanda a árvore objeto até encontrar a pasta **Tabelas** para o banco de dados para o qual você deseja importar os dados.
   b. Clique na pasta **Tabelas**. As tabelas são exibidas no painel de conteúdos no lado direito da janela.
   c. Clique com o botão direito na tabela que você está importando os dados e clique em **Spatial Extender** —> **Importar Dados Espaciais** no menu instantâneo. A janela Importar Dados Espaciais é aberta.
2. Na caixa **Nome de Arquivo**, especifique o caminho e o nome do arquivo que contém os dados a serem importados.
3. Use a caixa **Escopo do Commit** para especificar o número de registros que você deseja importar antes de cada commit. Por exemplo, para solicitar o commit do DB2 a cada 100 registros, especifique o número 100.

   Dica: Caso você queira o DB2 para emitir somente um commit após todos os registros serem processados, especifique zero.
4. Especifique a coluna que conterá os dados espaciais que você está importando.
   - Se a coluna não existe ainda na tabela, use a caixa **Coluna de Camada** para digitar um nome para a coluna.
   - Se a coluna já existe, use a caixa **Coluna de Camada** para selecionar ou digitar o nome da coluna.
5. Use a caixa **Nome de referência espacial** para especificar qual sistema de referência espacial deve ser associado aos dados importados.
   - Se estiver incluindo uma coluna em uma tabela, digite ou selecione o nome do sistema de referência espacial.
   - Se os dados importados estiverem planejados para uma coluna existente, deixe a caixa **Nome de referência espacial** como está. Ela exibe o nome do sistema de referência espacial padrão.
6. No campo **Arquivo de Exceção**, especifique o caminho e nome para um novo arquivo no qual os registros que falharem durante a importação sejam coletados. Mais tarde, você pode alterar estes registros e importá-los deste arquivo.

   O DB2 Spatial Extender criará este arquivo; não especifique um que já exista.

7. Clique em **OK** para importar os dados. Também, se você especificou uma coluna que não existe ainda, esta coluna será criada e registrada como uma camada. Complementarmente, o arquivo de exceção que você especificou será criado.

Exportando dados para um arquivo shape

Esta seção oferece uma visão geral das etapas para exportar dados espaciais para um arquivo shape. A visão geral está seguida por detalhes de como completar cada etapa.

Para encontrar que autorização é requerida para executar estas etapas, consulte "Autorização" na página 85.

Visão geral das etapas para exportar dados a partir de um arquivo shape:

1. Abra a janela Exportar Dados Espaciais
2. Especifique a coluna que contém os dados espaciais a serem exportados.
3. Caso deseja exportar um subconjunto de linhas de dados, identifique este subconjunto para o DB2 Spatial Extender.
4. Especifique o caminho e o nome do arquivo para o qual está exportando os dados.
5. Instrua o DB2 Spatial Extender para exportar os dados.

Detalhe das etapas para exportar dados para um arquivo shape:

1. Abra a janela Exportar Dados Espaciais
   a. A partir da janela Centro de Controle, expanda a árvore objeto até encontrar a pasta **Tabelas** ou **Views** no banco de dados que contém os dados espaciais:
   b. Clique na pasta **Tabelas** ou **Views**. Tabelas ou views são exibidas no painel de conteúdos no lado direito da janela.
   c. Clique com o botão direito na tabela ou view que contém os dados a serem exportados e clique em **Spatial Extender** —> **Exportar Dados Espaciais** no menu instantâneo. A janela Exportar Dados Espaciais é aberta.
2. No campo **Coluna de Camadas**, especifique o nome da coluna que contém os dados para serem exportados.
3. Caso queira exportar um subconjunto de linhas da tabela, use a caixa **cláusula WHERE** para digitar uma cláusula WHERE que especifica o critério para as linhas que deseja. Nesta cláusula, você pode referenciar apenas colunas na tabela ou view da qual esteja exportando dados.

   Digite somente os critérios. Omita a palavra-chave WHERE. Por exemplo, se a tabela ou view possui uma coluna STATE, e você deseja fazer o geocode apenas para as linhas que contenham o valor MA nesta coluna, digite:

   ```
   STATE='MA'
   ```

4. No campo **Nome de arquivo**, especifique o caminho e o nome do arquivo para o qual esteja exportando os dados.
5. Clique em **OK** para exportar os dados.

Capítulo 6. Criando índices espaciais

Este capítulo explica como usar o Centro de Controle para criar um índice para seus dados espaciais.

Após preencher as colunas espaciais com seus dados, você está pronto para criar um índice espacial. Estruturas de indexação comuns, como B tree, realizam ordenações lineares monodimensionais em dados da tabela. Os dados da tabela que foram habilitados para operações espaciais não são armazenados como uma única entrada, mas são bidimensionais. Por exemplo, as geometrias espaciais como polígonos consistem em vários valores de coordenadas em uma coluna ou camada espacial. Como um índice B tree não pode manipular tipos de dados espaciais, o DB2 Spatial Extender criou uma tecnologia de indexação de propriedade conhecida como *índice de grade*. O índice de grade baseia-se no índice B tree, que foi aprimorado para manipular dados bidimensionais e realizar indexações em colunas espaciais. O índice de grade suporta três camadas e foi projetado para oferecer um bom desempenho em uma faixa grande de objetos, tamanhos e distribuições de dados. Para obter mais informações sobre índices espaciais, consulte "Capítulo 12. Índices espaciais" na página 117.

Para descobrir qual é a autorização necessária para criação de um índice espacial, consulte "Autorização" na página 80.

Utilizando o Centro de Controle para criar um índice espacial

Para criar um índice espacial utilizando o Centro de Controle:

1. Na árvore de objetos, selecione a pasta **Tabelas**. Todas as tabelas existentes são exibidas no painel de conteúdo.
2. A partir do painel de conteúdo, clique com o botão direito sobre a tabela para a qual deseja criar um índice, e clique em **Spatial Extender** —> **Índices Espaciais** no menu instantâneo. A janela Índices Espaciais aparece.
3. A partir da janela Índices Espaciais, clique em **Criar**. A janela Criar Índice Espacial aparece.
4. No campo **Nome**, digite o nome do novo índice espacial a ser criado.

   Nota: Não é necessário especificar um esquema. O DB2 Spatial Extender inclui o esquema automaticamente e cria um nome completamente qualificado para você.
5. No campo **Coluna da camada**, selecione a camada para a qual está criando um índice.

Uma camada é uma coluna espacial definida ou registrada para o DB2 Spatial Extender.

6. Nos campos **Tamanho da grade**, digite o valor do tamanho da grade que deseja atribuir a cada campo.

   Os níveis da grade, **Mais fino**, **Médio** e **Mais grosso**, são fornecidos aumentando-se o tamanho da célula. Conseqüentemente, o segundo nível deve ter um tamanho de célula maior do que o primeiro e o terceiro maior que o segundo.

Determinando os tamanhos de células da grade

A determinação do tamanho correto da grade é feito através de um processo de tentativa e erro. Recomenda-se que o tamanho da grade seja definido em relação ao tamanho aproximado do objeto sendo indexado. Tamanhos de grade muito pequenos ou muito grandes podem resultar em um desempenho mais lento. Tamanhos pequenos demais afetam a proporção chave/objeto durante uma pesquisa do índice. Neste caso, um número excessivo de chaves é criado e um número grande de candidatos é retornado. Para tamanhos de grade muito grandes, a pesquisa de índice inicial retorna um número pequeno de candidatos, porém o desempenho pode ficar mais lento durante a pesquisa final da tabela.

Para saber mais sobre a escolha de tamanhos de células de grade e o número de níveis de grade, consulte “Selecionando o tamanho da célula de grade” na página 124.

Capítulo 7. Recuperando e analisando informações espaciais

Após construir os índices espaciais as tabelas espaciais estão prontas para uso. Este capítulo discute temas relacionados para recuperar e analisar dados espaciais. Ele contém uma visão geral de vários métodos de recuperação e fornece exemplo de consultas de tabela que usam funções espaciais.

Métodos de execução de análise espacial

Você pode executar análises espaciais utilizando SQL e funções espaciais com qualquer um dos seguintes ambientes de programação:

- Um geobrowser (por exemplo, ESRI's ArcExplorer).

  Para mais informações sobre o uso do ArcExplorer, consulte o *Using ArcExplorer*, que está disponível no site Web ESRI em http://www.esri.com.
- Instruções SQL interativas.
- Aplicações desenvolvidas pelo usuário (por exemplo, ODBC, JDBC, e SQL incorporadas).

As aplicações podem ser lançadas a partir do Centro de Comando do DB2, da Janela de Comandos do DB2 ou do processador da linha de comandos.

Construindo uma consulta espacial

Esta seção discute a construção de consultas espaciais que usam funções e predicados espaciais.

Funções espaciais e SQL

O DB2 Spatial Extender inclui funções que executam várias operações nos dados espaciais. Os exemplos nesta seção mostram como usar as funções espaciais para construir suas próprias consultas espaciais.

A Tabela 3 fornece uma lista das funções espaciais e os tipos de operações que elas podem executar.

Tabela 3. Operações e funções espaciais

Tipo de função	Exemplo de operação
Cálculo	Cálculo da distância entre 2 pontos
Comparação	Localiza todos os clientes localizados dentro de uma zona de inundação
Permuta de dados	Converte dados nos formatos suportados

Tabela 3. Operações e funções espaciais (continuação)

Tipo de função	Exemplo de operação
Transformação	Inclui um raio de cinco milhas em um ponto

Para obter mais informações sobre as funções espaciais, consulte “Capítulo 13. Geometrias e funções espaciais associadas” na página 127 e “Capítulo 14. Funções espaciais para consultas SQL” na página 163.

Exemplo 1: Comparação

A seguinte consulta localiza a distância média do cliente a cada estabelecimento. As funções espaciais usadas neste exemplo são ST_Distance e ST_Within.

```
SELECT s.id, AVG(db2gse.ST_Distance(c.location,s.location))
FROM customers c, stores s
WHERE db2gse.ST_Within(c.location,s.zone)=1
GROUP BY s.id
```

Exemplo 2: Permuta de dados

A seguinte consulta procura as localizações de clientes que vivem na Área da Baía de São Francisco (San Francisco Bay Area). As funções espaciais utilizadas neste exemplo são ST_AsText (permuta de dados) e ST_Within. ST_AsText converte os dados espaciais na coluna c.location no formato OGC TEXT.

```
SELECT db2gse.ST_AsText(c.location,cordref(1))
FROM customers c
WHERE db2gse.ST_Within(c.location,:BayArea)=1
```

Exemplo 3: Cálculo

A seguinte consulta procura todas as ruas maiores que 10.5 milhas. A função espacial utilizada neste exemplo é a ST_Length.

```
SELECT s.name,s.id
FROM street s
WHERE db2gse.ST_Length(s.path) > 10.5
```

Exemplo 4: Transformação

Esta consulta localiza os clientes que vivem dentro de uma área de inundação ou dentro de 2 milhas do limite da zona de inundação. As funções espaciais utilizadas neste exemplo são ST_Buffer (transformação) e ST_Within.

```
SELECT c.name,c.phoneNo,c.address
FROM customers c
WHERE db2gse.ST_Within(c.location,ST_Buffer(:floodzone,2))=1
```

Predicados espaciais e SQL

Um grupo especializado de funções espaciais que são chamados predicados espaciais podem aprimorar o desempenho da consulta. Predicados espaciais, tais como ST_Overlaps, que compara dois polígonos para ver se eles se sobrepõem, pode ser uma execução custosa por problemas de requisitos de memória e tempo. Portanto, as técnicas de otimizações para minimizar o custo

de execução são muito importantes. O otimizador de consultas do DB2 utiliza o índice espacial para aprimorar o desempenho da consulta quando você utiliza os predicados espaciais de acordo com as regras descritas mais adiante nesta seção. Para mais informações sobre os predicados espaciais, consulte "Funções do predicado" na página 141. Os predicados espaciais usados para explorar o índice espacial são:

- ST_Contains
- ST_Crosses
- ST_Disjoint
- ST_Distance
- ST_Envelope
- ST_Equals
- ST_Intersects
- ST_Overlaps
- ST_Touches
- ST_Within

Para uma lista completa de todas as funções e predicados espaciais, consulte "Capítulo 14. Funções espaciais para consultas SQL" na página 163.

Regras para exploração de índice

As seguintes regras se aplicam caso deseje otimizar consultas espaciais utilizando predicados espaciais:

- O predicado deve ser utilizado na cláusula WHERE.
- O predicado deve estar do lado esquerdo da comparação. Por exemplo:

  ```
  WHERE db2gse.ST_Within(c.location,:BayArea)=1
  ```

- Comparações de igualdade devem usar o inteiro constante 1.

  ```
  WHERE db2gse.ST_Within(c.location,:BayArea)=1
  ```

- Deve ser uma coluna espacial usada no predicado como a pesquisa de destino, e deve ser um índice espacial criado naquela coluna.

Exemplos da exploração do índice

A Tabela 4 mostra as maneiras corretas e incorretas de criar consultas espaciais para explorar o índice espacial.

Tabela 4. Regras para exploração de índice

Consulta espaciais	Regras violadas

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Within(c.location,:BayArea)=1`
	Nenhuma condição é violada neste exemplo.

Tabela 4. Regras para exploração de índice (continuação)

Consulta espaciais	Regras violadas

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Distance(c.location,:SanJose)<10`
	Nenhuma condição é violada neste exemplo.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Length(c.location)>10`
	O predicado deve ser utilizado na cláusula WHERE. (ST_Length é uma função espacial, porém *não* um predicado.)

`SELECT *`
`FROM customers c`
`WHERE 1=db2gse.ST_Within(c.location,:BayArea)`
	O predicado deve estar do lado esquerdo da comparação.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Within(c.location,:BayArea)=2`
	Comparações de igualdade devem usar o inteiro constante 1.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Within(:SanJose,:BayArea)=1`
	Deve ser uma coluna espacial usada no predicado como a pesquisa de destino, e deve ser um índice espacial criado naquela coluna. (SanJose e BayArea não são colunas espaciais e portanto, não podem ter um índice espacial associado às mesmas.)

Capítulo 8. Gravando aplicações para DB2 Spatial Extender

Este capítulo explica como usar o programa de amostra do DB2 Spatial Extender para escrever aplicações para trabalhar e personalizar informações espaciais. Estão incluídos os seguintes tópicos:

- Utilizando o programa de amostra
- As etapas do programa de amostra

Utilizando o programa de amostra

O programa de amostra DB2 Spatial Extender torna a programação de aplicação mais fácil. Com o programa de amostra, você pode:

- Automatizar as rotinas de procedimentos espaciais
- Cortar e colar o código de amostra em suas aplicações
- Entender as etapas tipicamente requeridas para criar e manter um banco de dados ativado espacialmente

Use o programa de amostra para codificar tarefas complexas para o DB2 Spatial Extender, por exemplo para escrever aplicações que usam interface de banco de dados para chamar os procedimentos armazenados do DB2 Spatial Extender. A partir do programa de amostra, você pode copiar e personalizar suas aplicações. Caso não esteja familiarizado com as etapas de programação do DB2 Spatial Extender, você pode executar o programa de amostra, que exibe a você cada etapa em detalhe. Primeiro, porém, você deve criar o programa de amostra. Faça isso com o makefile de amostra. Para obter instruções sobre a criação e a execução do programa de amostra, consulte “Verificando a instalação” na página 20.

As etapas do programa de amostra

A Tabela 5 na página 60 exibe as etapas do programa de amostra, os procedimentos armazenados associados e uma descrição de cada etapa. As funções C para chamar os procedimentos armazenados são exibidas na coluna Ação da Tabela 5 na página 60 e estão entre parêntesis. Para obter mais informações sobre os procedimentos armazenados, consulte “Capítulo 9. Procedimentos armazenados” na página 69. O programa de amostra está baseado em cenários que estão introduzidos em “Cenário: Uma companhia de seguros atualiza seu GIS” na página 12.

Tabela 5. Programa de amostra do DB2 Spatial Extender

Etapas do programa de amostra	Ação	Descrição
Ativar / desativar o banco de dados espacial	1. Ativar o banco de dados espacial (gseEnableDB) 2. Desativar o banco de dados espacial (gseDisableDB) 3. Ativar o banco de dados espacial (gseEnableDB)	1. Esta é a primeira etapa necessária para utilizar o DB2 Spatial Extender. Um banco de dados que foi ativado para operações espaciais possui um conjunto de tipos espaciais, um conjunto de funções espaciais, um conjunto de predicados espaciais, um novo tipo de índice e um conjunto de administração de tabelas e views. 2. Esta etapa é normalmente executada quando você tem capacidades espaciais ativadas para o banco de dados errado. Ao desativar um banco de dados, você remove um conjunto de tipos espaciais, um conjunto de funções espaciais, um conjunto de predicados espaciais, um novo tipo de índice e um conjunto de administração de tabelas e views. **Nota:** O desativar banco de dados falhará se houver objetos criados que dependem dos objetos criados pelo ativar procedimento de banco de dados. Por exemplo, ao criar uma tabela com uma coluna espacial do tipo ST_Point causará a falha do desativar banco de dados. Isto ocorre porque a tabela depende do ST_Point que é destinada a ser eliminada pelo desativar procedimento de banco de dados. 3. O mesmo que 1.

Tabela 5. Programa de amostra do DB2 Spatial Extender (continuação)

Etapas do programa de amostra	Ação	Descrição
Registrar sistemas de referência espaciais	1. Registrar o sistema de referência espacial para a coluna LOCATION da tabela CUSTOMERS (gseEnableSref) 2. Registrar o sistema de referência espacial para a coluna LOCATION da tabela OFFICES (gseEnableSref) 3. Desregistrar o sistema de referência espacial para a coluna LOCATION da tabela OFFICES (gseDisableSref) 4. Re-registrar o sistema de referência espacial para as colunas ZONE da tabela OFFICES (gseEnableSref)	1. Esta etapa define um novo sistema de referência espacial (SRS) a ser utilizado para interpretar os dados espaciais da tabela CUSTOMERS. Um sistema de referencia espacial inclui dados de geometria de uma maneira que podem ser armazenados em uma coluna de um banco de dados espacial. Após o SRS ser registrado para uma camada específica, as coordenadas aplicáveis para a camada podem ser armazenadas na coluna da tabela CUSTOMERS associada. 2. Esta etapa define um novo sistema de referência espacial (SRS) a ser utilizado para interpretar os dados espaciais da camada OFFICES.Cada camada de tabela deve ter um SRS definido para a mesma. As camadas da tabela OFFICES podem requerer um SRS associado diferente da camada de tabela CUSTOMERS. 3. Esta etapa é executada se você especifica um parâmetro SRS errado para a camada ou coluna espacial. Ao desregistrar um SRS para a camada da tabela OFFICES, você remove a definição com os seus parâmetros associados. 4. Esta etapa define um novo sistema de referência espacial (SRS) a ser utilizado para interpretar os dados espaciais da camada OFFICES.

Tabela 5. Programa de amostra do DB2 Spatial Extender (continuação)

Etapas do programa de amostra	Ação	Descrição
Criar as tabelas espaciais	1. Alterar a tabela CUSTOMERS incluindo a coluna LOCATION (gseSetupTables) 2. Criar a tabela OFFICES (gseSetupTables)	1. A tabela CUSTOMERS representa dados de negócios que foram armazenados no banco de dados por vários anos. A instrução ALTER TABLE inclui uma nova coluna (LOCATION) do tipo ST_Point. Esta coluna será preenchida pela geocodificação das colunas de endereço em uma etapa subseqüente. 2. A tabela OFFICES representa, entre outros dados, uma zona de vendas para cada escritório de uma companhia de seguros. A tabela inteira será preenchida com os dados de atributo de um banco de dados não-DB2 em uma etapa subseqüente. Esta etapa envolve a importação de dados de atributo para a tabela OFFICES a partir de um arquivo SHAPE.
Registrar as camadas espaciais	1. Registrar a coluna LOCATION na tabela CUSTOMERS como uma camada (gseRegisterLayer) 2. Registrar a coluna ZONE da tabela OFFICES como uma camada (gseRegisterLayer)	Estas etapas registram as colunas LOCATION e ZONE como camadas para o DB2 Spatial Extender. Antes que uma coluna espacial seja preenchida ou acessada pelos utilitários do DB2 Spatial Extender (por exemplo, o geocoder), você precisa registrá-lo como uma camada.

Tabela 5. Programa de amostra do DB2 Spatial Extender (continuação)

Etapas do programa de amostra	Ação	Descrição
Preencher as camadas espaciais	1. Fazer o geocode dos dados de endereços para a coluna LOCATION da tabela CUSTOMERS (gseRunGC) 2. Carregar a tabela OFFICES usando o modo anexar (gseImportShape) 3. Carregar a tabela HAZARD_ZONE usando o modo criar (gseImportShape)	1. Esta etapa executa a geocodificação através da chamada do utilitário geocoder. A geocodificação em batch é normalmente executada quando uma porção significativa da tabela precisa ser geocodificada ou re-geocodificada. 2. Esta etapa carrega a tabela OFFICES com os dados espaciais existentes no formulário de um arquivo SHAPE. Por causa da existência da tabela OFFICES e da camada OFFICES/ZONE estar registrada, o utilitário de carga irá anexar os novos registros a uma tabela existente. 3. Esta etapa carrega a camada HAZARD_ZONE com os dados espaciais existentes no formulário de um arquivo SHAPE. Pelo fato da tabela e da camada não existirem, o utilitário de carga criará a tabela e registrará a camada antes que os dados sejam carregados.
Ativar índices espaciais	1. Ativar o índice espacial para a coluna LOCATION da tabela CUSTOMERS (gseEnableIdx) 2. Ativar o índice espacial para a coluna ZONE da tabela OFFICES (gseEnableIdx) 3. Ativar o índice espacial para a coluna LOCATION da tabela OFFICES (gseEnableIdx) 4. Ativar o índice espacial para a coluna BOUNDRY da tabela HAZARD_ZONE (gseEnableIdx)	Estas etapas ativam o índice espacial para as tabelas CUSTOMERS, OFFICES e HAZARD_ZONE.
Ativar a geocodificação automática	1. Ativar a geocodificação automática para as colunas LOCATION e ADDRESS da tabela CUSTOMERS (gseEnableAutoGC)	Esta etapa torna a chamada do geocoder automática. A utilização da geocodificação automática causa às colunas LOCATION e ADDRESS da tabela CUSTOMERS a sincronização com cada outra para operações de inclusão e atualização subseqüentes.

Tabela 5. Programa de amostra do DB2 Spatial Extender (continuação)

Etapas do programa de amostra	Ação	Descrição
Incluir / atualizar a tabela CUSTOMERS	1. Inserir alguns registros com uma rua diferente (gseInsDelUpd) 2. Atualizar alguns registros com um novo endereço (gseInsDelUpd)	Estas etapas demonstram uma inclusão e atualização na coluna LOCATION da tabela CUSTOMERS. Uma vez ativada a geocodificação automática, as informações a partir da coluna ADDRESS são automaticamente geocodificadas quando ela é incluída ou atualizada na coluna LOCATION. Este processo foi ativado na etapa anterior.
Desativar a geocodificação automática	1. Desativar a geocodificação automática para a camada CUSTOMERS (gseDisableAutoGC) 2. Desativar o índice espacial para a camada CUSTOMERS (gseDisableIdxCustomersLayer)	Estas etapas desativam a chamada automática do geocoder e do índice espacial em preparação para a próxima etapa (a próxima etapa envolve a re-geocodificação da tabela CUSTOMERS completa). Caso esteja carregando uma grande tabela de geodados, é recomendado que você desative o índice espacial antes de carregar os dados, ativando-o após a conclusão da carga.
Re-geocodificar a tabela CUSTOMERS	1. Fazer o geocode da camada CUSTOMERS novamente com um nível de precisão baixo – 90% ao invés de 100% (gseRunGC) 2. Reativar o índice espacial para a camada CUSTOMERS (gseEnableIdx) 3. Reativar a geocodificação automática com um nível de precisão baixo – 90% ao invés de 100% (gseEnableAutoGC)	Estas etapas executam o geocoder no modo batch novamente, re-ativa a geocodificação automática com um novo nível de precisão e re-ativa o índice espacial e a geocodificação automática. Esta ação é recomendada quando o administrador espacial informa uma taxa de falhas alta no processo de geocodificação. Se o nível de precisão está definido em 100%, ele pode falhar para o geocode de um endereço porque pode não encontrar um endereço correspondente nos dados de referência. Pela redução do nível de precisão, o geocoder tem uma chance maior de encontrar dados coincidentes. Após a tabela ser re-geocodificada no modo batch, ambos, a geocodificação automática e o índice espacial, são ativados novamente para facilitar a manutenção incremental do índice e da coluna espacial para inclusões e alterações subseqüentes.

Tabela 5. Programa de amostra do DB2 Spatial Extender (continuação)

Etapas do programa de amostra	Ação	Descrição
Criar uma view e registre suas colunas espaciais como camadas de view	1. Criar uma view, HIGHRISK_CUSTOMERS, baseada na junção das tabelas CUSTOMERS e HAZARD_ZONE (gseCreateView) 2. Registrar as colunas espaciais da view como camadas de view (gseRegisterLayer)	Estas etapas criam uma view e registram suas colunas espaciais como camadas de view.
Executar análise espacial	1. Localizar a distância média do cliente para cada escritório (ST_Within, ST_Distance) 2. Localiza a renda média do cliente e o premio para cada escritório (ST_Within) 3. Localizar clientes que não estão cobertos por um escritório existente (ST_Within) 4. Localizar o número de zonas de risco que cada escritório sobrepõe (ST_Overlaps) 5. Localizar o escritório mais próximo a partir de uma localização de cliente particular assumindo que o escritório esteja localizado no centro da zona do escritório (ST_Distance, ST_Centroid) 6. Localizar os clientes cuja localização esteja próxima ao limite de uma zona de risco particular (ST_Buffer, ST_Overlaps) 7. Localizar os clientes de risco alto que estão cobertos por um escritório em particular (Todas as etapas utilizam a gseRunSpatialQueries)	Estas etapas executam análises espaciais utilizando os predicados espaciais e funções na linguagem SQL do DB2. O otimizador de consultas do DB2 explora o índice espacial nas colunas espaciais para aprimorar o desempenho das consultas sempre que possível.

Tabela 5. Programa de amostra do DB2 Spatial Extender (continuação)

Etapas do programa de amostra	Ação	Descrição
Exportar as camadas espaciais para dentro dos arquivos	Exportar as camadas highRiskCustomers (gseExportShape)	A etapa mostra um exemplo de exportação dos resultados de sua consulta para um arquivo SHAPE. A exportação dos resultados de consulta para outro formato de arquivo permite que a informação possa ser utilizada por uma terceira ferramenta (por exemplo, ESRI ArcInfo).

Parte 2. Material de referência

Capítulo 9. Procedimentos armazenados

Este capítulo documenta os procedimentos armazenados que permitem a construção de um sistema de informações geográficas com o DB2 Spatial Extender. Quando o DB2 Spatial Extender é ativado e usado a partir do Centro de Controle, estes procedimentos armazenados são chamados implicitamente. Por exemplo, quando você clica em **OK** a partir de uma janela do DB2 Spatial Extender, o DB2 chama os procedimentos armazenados associados a essa janela. Como alternativa, você pode chamar os procedimentos armazenados num programa da aplicação. É aconselhável incluir o arquivo de cabeçalho, db2gse.h, em tal programa. Este arquivo contém as definições de macro para as constantes que você atribuir aos parâmetros dos procedimentos armazenados. No AIX, ele está armazenado no diretório $DB2INSTANCE/sqllib/include/. No Windows NT, está armazenado no diretório %DB2PATH%\include\.

Atenção:
Todas as constantes da cadeia de caracteres dos parâmetros de entrada dos procedimentos armazenados são sensíveis a maiúsculas e minúsculas. Para descobrir quais parâmetros requerem essas constantes, consulte as tabelas deste capítulo.

Antes de chamar um procedimento armazenado, seja implícita ou explicitamente, você deve estar conectado ao banco de dados no qual o DB2 Spatial Extender está instalado. O primeiro procedimento armazenado que você utilizar será db2gse.gse_enable_db. Ele ativa o banco de dados para operações espaciais. Você poderá usar outros procedimentos armazenados somente após a ativação do banco de dados.

As implementações dos procedimentos armazenados estão arquivadas na biblioteca db2gse do servidor DB2 Spatial Extender.

As seguintes listas podem ser usadas para consulta dos procedimentos armazenados, seja pelos nomes ou pelas tarefas que podem realizar. A primeira lista apresenta os nomes:

- “db2gse.gse_enable_idx” na página 80
- “db2gse.gse_enable_sref” na página 82
- “db2gse.gse_export_shape” na página 85
- “db2gse.gse_import_sde” na página 87
- “db2gse.gse_import_shape” na página 89
- “db2gse.gse_register_gc” na página 91
- “db2gse.gse_register_layer” na página 93
- “db2gse.gse_run_gc” na página 99
- “db2gse.gse_unregist_gc” na página 101
- “db2gse.gse_unregist_layer” na página 102

A lista seguinte apresenta as tarefas que os procedimentos armazenados executam.

- Criação de um índice para uma coluna espacial (consulte “db2gse.gse_enable_idx” na página 80).
- Criação de um sistema de referência espacial (consulte “db2gse.gse_enable_sref” na página 82).
- Desativação de um geocoder para que não seja possível manter automaticamente colunas espaciais sincronizadas com suas colunas de atributos correspondentes (consulte “db2gse.gse_disable_autogc” na página 72).
- Desativação do suporte para operações espaciais num banco de dados (consulte “db2gse.gse_disable_db” na página 74).
- Eliminação de um sistema de referência espacial (consulte “db2gse.gse_disable_sref” na página 75).
- Ativação de um banco de dados para suportar operações espaciais (consulte “db2gse.gse_enable_db” na página 79).
- Ativação de um geocoder para manter automaticamente colunas espaciais sincronizadas com suas colunas de atributos correspondentes (consulte “db2gse.gse_enable_autogc” na página 76).
- Exportação de uma camada e sua tabela associada a um arquivo de shape (consulte “db2gse.gse_export_shape” na página 85).
- Importação de uma camada e sua tabela associada de um arquivo de transferência do ESRI_SDE (consulte “db2gse.gse_import_sde” na página 87).
- Importação de uma camada e sua tabela associada de um arquivo de shape (consulte “db2gse.gse_import_shape” na página 89).
- Registro de um geocoder diferente do padrão (consulte “db2gse.gse_register_gc” na página 91).
- Registro de uma coluna espacial como uma camada (consulte “db2gse.gse_register_layer” na página 93).

- Execução de um geocoder no modo batch (consulte “db2gse.gse_unregist_gc” na página 101).
- Cancelamento do registro de um geocoder diferente do padrão (consulte “db2gse.gse_unregist_layer” na página 102).
- Cancelamento do registro de uma camada (consulte “db2gse.gse_unregist_layer” na página 102).

Para obter informações sobre as seqüências nas quais é possível executar estas tarefas, consulte “Capítulo 1. Sobre o DB2 Spatial Extender” na página 3 e “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

db2gse.gse_disable_autogc

Use este procedimento armazenado para eliminar ou desativar temporariamente disparadores que mantêm uma coluna espacial sincronizada com suas colunas de atributos. Por exemplo, é aconselhável desativar os disparadores enquanto você geocodifica os valores na coluna ou colunas atributos no modo batch. Para mais informações, consulte “Sobre geocodificação” na página 41.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseDisableAutoGc no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

A ID de usuário com a qual esse procedimento armazenado foi chamado deve ter autorização na forma de uma autoridade, privilégio ou conjunto de privilégios, especificamente:

- Autoridade SYSADM ou DBADM no banco de dados que contém a tabela na qual estão definidos os disparadores que estão sendo eliminados ou desativados temporariamente.
- O privilégio CONTROL nesta tabela.
- Os privilégios ALTER, SELECT e UPDATE nesta tabela.

Parâmetros de entrada

Tabela 6. Parâmetros de entrada para o procedimento armazenado db2gse.gse_disable_autogc.

Nome	Tipo de dados	Descrição
operMode	SMALLINT	Indica se os disparadores deverão ser eliminados ou desativados temporariamente. Este parâmetro não pode ser nulo. **Comentário**: Para eliminar disparadores, use a macro GSE_AUTOGC_DROP. Para desativá-los temporariamente, use a macro GSE_AUTOGC_INVALIDATE. Para descobrir quais valores estão associados a estas macros, consulte o arquivo db2gse.h. No AIX, ele está armazenado no diretório $DB2INSTANCE/sqllib/include/. No Windows NT, está armazenado no diretório %DB2PATH%\include\.

Tabela 6. Parâmetros de entrada para o procedimento armazenado db2gse.gse_disable_autogc. (continuação)

Nome	Tipo de dados	Descrição
layerSchema	VARCHAR(30)	Nome do esquema ao qual pertence a tabela ou view especificados no parâmetro layerTable. Este parâmetro pode ser nulo. **Comentário:** Se você não fornecer um valor para o parâmetro layerSchema, ele assumirá a ID de usuário com que o procedimento armazenado db2gse.gse_disable_autogc é chamado.
layerTable	VARCHAR(128)	Nome da tabela na qual estão definidos os disparadores que você deseja eliminar ou desativar temporariamente. Este parâmetro não pode ser nulo.
layerColumn	VARCHAR(128)	Nome da coluna ativada espacialmente que é mantida pelos disparadores que você deseja eliminar ou desativar temporariamente. Este parâmetro não pode ser nulo.

Parâmetros de saída

Tabela 7. Parâmetros de saída para o procedimento armazenado db2gse.gse_disable_autogc.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_disable_db

Use este procedimento armazenado para remover recursos que permitem que o DB2 Spatial Extender armazene dados espaciais e suporte operações executadas nestes dados.

O objetivo deste procedimento armazenado é ajudá-lo a resolver problemas ou questões que surgirem após você ativar o banco de dados para operações espaciais, porém *antes* inclua qualquer coluna ou dados da tabela nele. Por exemplo, ser depois de ativar um banco de dados para operações espaciais, for decidido que DB2 Spatial Extender será usado para outro banco de dados. Contanto que você não tenha definido colunas espaciais ou importado dados espaciais, poderá chamar este procedimento armazenado para remover todos os recursos espaciais do primeiro banco de dados.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseDisableDB no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

A ID de usuário com a qual este procedimento armazenado é chamado deve ter autoridade SYSADM ou DBADM no banco de dados a partir do qual os recursos do DB2 Spatial Extender deverão ser removidos.

Parâmetros de saída

Tabela 8. Parâmetros de saída para o procedimento armazenado db2gse.gse_disable_db.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_disable_sref

Use este procedimento armazenado para eliminar um sistema de referência espacial. Quando este procedimento armazenado estiver processado, as informações sobre o sistema de referência espacial serão removidas da view do catálogo DB2GSE.SPATIAL_REF_SYS. Para obter informações sobre esta view, consulte “DB2GSE.SPATIAL_ REF_SYS” na página 115.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseDisableSref no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

Nenhuma é necessária.

Parâmetro de entrada

Tabela 9. Parâmetros de entrada para o procedimento armazenado db2gse.gse_disable_sref.

Nome	Tipo de dados	Descrição
srId	INTEGER	Identificador numérico do sistema de referência espacial que será eliminado. Este parâmetro não pode ser nulo.

Parâmetros de saída

Tabela 10. Parâmetros de saída para o procedimento armazenado db2gse.gse_disable_sref.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

Restrição

Antes de eliminar um sistema de referência espacial, você deverá cancelar o registro de qualquer camada que o utilize.Se tais camadas permanecerem sem registro, a solicitação de eliminação do sistema de referência espacial será rejeitada.

db2gse.gse_enable_autogc

Use este procedimento armazenado para:

- Criar disparadores que irão manter uma coluna espacial sincronizada com as colunas de atributos associadas. Sempre que os valores forem inseridos ou atualizados na coluna atributo, um disparador irá chamar um geocoder registrado para geocodificar os valores inseridos ou atualizados e posicionar os dados resultantes na coluna espacial.
- Reativar os disparadores após sua desativação temporária.
- Estabeleçer qual função será usada para geocodificar os valores inseridos e atualizados.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseEnableAutoGC no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

A ID de usuário com a qual esse procedimento armazenado foi chamado deve ter autorização na forma de uma autoridade, privilégio ou conjunto de privilégios, especificamente:

- Autoridade SYSADM ou DBADM no banco de dados que contém a tabela na qual estão definidos os disparadores criados pelo procedimento armazenado.
- O privilégio CONTROL nesta tabela.
- Os privilégios ALTER, SELECT e UPDATE nesta tabela.

Parâmetros de entrada

Tabela 11. Parâmetros de entrada para o procedimento armazenado db2gse.gse_enable_autogc.

Nome	Tipo de dados	Descrição
operMode	SMALLINT	Valor que indica se os disparadores que iniciam a geocodificação serão criados pela primeira vez ou serão reativados após a desativação temporária. Este parâmetro não pode ser nulo. **Comentário:** Para criar os disparadores, use a macro GSE_AUTOGC_CREATE. Para reativá-los, use a macro GSE_AUTOGC_RECREATE. Para descobrir quais valores estão associados a estas macros, consulte o arquivo db2gse.h. No AIX, ele está armazenado no diretório $DB2INSTANCE/sqllib/include/. No Windows NT, está armazenado no diretório %DB2PATH%\include\.

Tabela 11. Parâmetros de entrada para o procedimento armazenado db2gse.gse_enable_autogc. (continuação)

Nome	Tipo de dados	Descrição
layerSchema	VARCHAR(30)	Nome do esquema ao qual pertence a tabela especificada no parâmetro layerTable. Este parâmetro pode ser nulo. **Comentário:** Se você não fornecer um valor para o parâmetro layerSchema, ele assumirá a ID de usuário com que o procedimento armazenado db2gse.gse_enable_autogc é chamado.
layerTable	VARCHAR(128)	Nome da tabela em que serão operados os disparadores criados ou reativados por este procedimento armazenado. Este parâmetro não pode ser nulo.
layerColumn	VARCHAR(128)	Nome da coluna espacial que deverá ser mantida pelos disparadores que este procedimento armazenado cria ou reativa. Este parâmetro não pode ser nulo.
gcId	INTEGER	Identificador do geocoder que será chamado pelos disparadores de inserção e atualização que este procedimento armazenado cria ou reativa. Este parâmetro não pode ser nulo se o parâmetro operMode for definido como GSE_AUTOGC_CREATE. Ele poderá ser nulo se operMode estiver definido como GSE_AUTOGC_RECREATE.
precisionLevel	INTEGER	O grau em que os dados fonte devem coincidir dados de referência correspondentes para que o geocoder processe os dados fonte com êxito. Este parâmetro não pode ser nulo se o parâmetro operMode for definido como GSE_AUTOGC_CREATE. Ele poderá ser nulo se operMode estiver definido como GSE_AUTOGC_RECREATE. **Comentário:** O nível de precisão pode variar de 1 a 100%.

Tabela 11. Parâmetros de entrada para o procedimento armazenado db2gse.gse_enable_autogc. (continuação)

Nome	Tipo de dados	Descrição
vendorSpecific	VARCHAR(256)	Informações técnicas fornecidas pelo fornecedor; por exemplo, o caminho e nome de um arquivo que o fornecedor usa para definir parâmetros. Este parâmetro não pode ser nulo se o parâmetro operMode for definido como GSE_AUTOGC_CREATE. Ele poderá ser nulo se operMode estiver definido como GSE_AUTOGC_RECREATE.

Parâmetros de saída

Tabela 12. Parâmetros de saída para o procedimento armazenado db2gse.gse_enable_autogc.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

Restrições

- O parâmetro layerColumn deve fazer referência a uma coluna que tenha sido registrada como camada da tabela.
- Se o parâmetro operMode estiver definido em GSE_AUTOGC_CREATE, você deverá atribuir um identificador de um geocoder registrado ao parâmetro gcId.

db2gse.gse_enable_db

Use este procedimento armazenado para fornecer a um banco de dados os recursos de que ele precisa para armazenar dados espaciais e suportar operações. Estes recursos incluem tipos de dados espaciais, um tipo de índice espacial, tabelas e views do catálogo, funções fornecidas e outros procedimentos armazenados.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseEnableDB no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

A ID de usuário com a qual o procedimento armazenado é chamado deve ter autoridade SYSADM ou DBADM no banco de dados que está sendo ativado.

Parâmetros de saída

Tabela 13. Parâmetros de saída para o procedimento armazenado db2gse.gse_enable_db.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_enable_idx

Use este procedimento armazenado para criar um índice para uma coluna espacial.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseEnableIdx no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

A ID de usuário com a qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade SYSADM ou DBADM no banco de dados que contém a tabela na qual o índice ativado deverá ser usado.
- O privilégio CONTROL ou INDEX nesta tabela.

Parâmetros de entrada

Tabela 14. Parâmetros de entrada para o procedimento armazenado db2gse.gse_enable_idx.

Nome	Tipo de dados	Descrição
layerSchema	VARCHAR(30)	Nome do esquema ao qual pertence a tabela especificada no parâmetro layerTable. Este parâmetro pode ser nulo. **Comentário:** Se você não fornecer um valor para o parâmetro layerSchema, ele assumirá a ID de usuário com que o procedimento armazenado db2gse.gse_enable_idx é chamado.
layerTable	VARCHAR(128)	Nome da tabela na qual deverá ser definido o índice que você está criando. Este parâmetro não pode ser nulo.
layerColumn	VARCHAR(128)	Nome da coluna ativada espacialmente que deverá ser pesquisada com a ajuda do índice que está sendo criado. Este parâmetro não pode ser nulo.
indexName	VARCHAR(128)	Nome do índice que deverá ser criado. Este parâmetro não pode ser nulo. **Comentário:** Não especifique um nome de esquema. O DB2 Spatial Extender automaticamente atribui o índice ao esquema referido pelo parâmetro layerSchema.

Tabela 14. Parâmetros de entrada para o procedimento armazenado db2gse.gse_enable_idx. (continuação)

Nome	Tipo de dados	Descrição
gridSize1	DOUBLE	Número que indica qual deve ser a granulosidade da melhor grade de índice. Este parâmetro não pode ser nulo.
gridSize2	DOUBLE	Número que indica (1) que não deve haver segunda grade para este índice ou (2) qual deve ser a granulosidade da segunda grade. Este parâmetro pode ser nulo. **Comentário:** Se a segunda grade não deve existir, especifique 0. Se deseja uma segunda grade, ela deverá ser menos granulosa que a grade indicada por gridSize1.
gridSize3	DOUBLE	Número que indica (1) que não deve haver terceira grade para este índice ou (2) qual deve ser a granulosidade da terceira grade. Este parâmetro pode ser nulo. **Comentário:** Se a terceira grade não deve existir, especifique 0. Se deseja uma terceira grade, ela deverá ser menos granulosa que a grade indicada por gridSize2.

Parâmetros de saída

Tabela 15. Parâmetros de saída para o procedimento armazenado db2gse.gse_enable_idx.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_enable_sref

Use este procedimento armazenado para especificar como os números negativos e decimais em um sistema de coordenadas específico deverão ser convertidos em inteiros positivos para que o DB2 Spatial Extender possa armazená-los. Suas especificações são denominadas coletivamente *sistema de referência espacial*. Quando este procedimento armazenado estiver processado, as informações sobre o sistema de referência espacial serão incluídas na view do catálogo DB2GSE.SPATIAL_REF_SYS. Para obter informações sobre esta view, consulte “DB2GSE.SPATIAL_ REF_SYS” na página 115.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseEnableSref no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

Nenhuma é necessária.

Parâmetros de entrada

Tabela 16. Parâmetros de entrada para o procedimento armazenado db2gse.gse_enable_sref.

Nome	Tipo de dados	Descrição
srId	INTEGER	Um identificador numérico para o sistema de referência espacial. Este parâmetro não pode ser nulo. **Comentário**: Este identificador deve ser exclusivo dentro do banco de dados ativado espacialmente.
srName	VARCHAR(64)	Descrição breve do sistema de referência espacial. Este parâmetro não pode ser nulo. **Comentário**: Esta descrição deve ser exclusiva dentro do banco de dados ativado espacialmente.
falsex	DOUBLE	Um número que, quando subtraído de um valor negativo da coordenada X, deixa um número não-negativo (ou seja, um número positivo ou zero). Este parâmetro não pode ser nulo.

Tabela 16. Parâmetros de entrada para o procedimento armazenado db2gse.gse_enable_sref. (continuação)

Nome	Tipo de dados	Descrição
falsey	DOUBLE	Um número que, quando subtraído de um valor negativo da coordenada Y, permite um número não-negativo (ou seja, um número positivo ou zero). Este parâmetro não pode ser nulo.
xyunits	DOUBLE	Um número que, quando multiplicado por uma coordenada X decimal ou uma coordenada Y decimal, produz um inteiro que pode ser armazenado como um item de dados de 32 bits. Este parâmetro não pode ser nulo.
falsez	DOUBLE	Um número que, quando subtraído de um valor negativo da coordenada Z, permite um número não-negativo (ou seja, um número positivo ou zero). Este parâmetro não pode ser nulo.
zunits	DOUBLE	Um número que, quando multiplicado por uma coordenada Z decimal, produz um inteiro que pode ser armazenado como um item de dados de 32 bits. Este parâmetro não pode ser nulo.
falsem	DOUBLE	Um número que, quando subtraído de uma medida negativa, permite um número não-negativo (ou seja, um número positivo ou zero). Este parâmetro não pode ser nulo.
munits	DOUBLE	Um número que, quando multiplicado por uma medida decimal, produz um inteiro que pode ser armazenado como um item de dados de 32 bits. Este parâmetro não pode ser nulo.
scId	INTEGER	Identificador numérico do sistema de coordenadas do qual é derivado o sistema de referência espacial. Para descobrir o que é um identificador numérico do sistema de coordenadas, consulte a view do catálogo DB2GSE.COORD_REF_SYS “DB2GSE.COORD_REF_SYS” na página 113. Este parâmetro não pode ser nulo.

Parâmetros de saída

Tabela 17. Parâmetros de saída para o procedimento armazenado db2gse.gse_enable_sref.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_export_shape

Use este procedimento armazenado para exportar uma camada e sua tabela associada a um arquivo de shape ou criar um novo arquivo de shape e exportar uma camada e sua tabela associada para este novo arquivo.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseExportShape no programa de amostra. Para obter informações sobre este programa, consulte "Capítulo 8. Gravando aplicações para DB2 Spatial Extender" na página 59.

Autorização

A ID de usuário com a qual este procedimento armazenado é chamado deve ter o privilégio SELECT na tabela que deverá ser exportada.

Parâmetros de entrada

Tabela 18. Parâmetros de entrada para o procedimento armazenado db2gse.gse_export_shape.

Nome	Tipo de dados	Descrição
layerSchema	VARCHAR(30)	Nome do esquema ao qual pertence a tabela especificada no parâmetro layerTable. Este parâmetro pode ser nulo. **Comentário:** Se você não fornecer um valor para o parâmetro layerSchema, ele assumirá a ID de usuário com que o procedimento armazenado db2gse.gse_export_shape é chamado.
layerTable	VARCHAR(128)	Nome da tabela que você está exportando. Este parâmetro não pode ser nulo.
layerColumn	VARCHAR(30)	Nome da coluna que foi registrada como sendo a camada que você está exportando. Este parâmetro não pode ser nulo.
fileName	VARCHAR(128)	Nome do arquivo de shape para o qual a camada especificada deverá ser exportada. Este parâmetro não pode ser nulo.
whereClause	VARCHAR(1024)	O corpo da cláusula SQL WHERE. Define uma restrição ao conjunto de registros que serão geocodificados. A cláusula pode referenciar qualquer coluna de atributo na tabela que você está exportando. Este parâmetro pode ser nulo.

Parâmetros de saída

Tabela 19. Parâmetros de saída para o procedimento armazenado db2gse.gse_export_shape.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

Restrição

É possível exportar somente uma camada por vez.

db2gse.gse_import_sde

Use este procedimento armazenado para importar um arquivo de transferência SDE para um banco de dados que tenha sido ativado para operações espaciais. O procedimento armazenado pode operar de uma destas duas formas:

- Se o arquivo de transferência SDE destinar-se a uma tabela existente que tenha uma coluna de camada registrada, o DB2 Spatial Extender carregará a tabela com os dados do arquivo.
- Caso contrário, DB2 Spatial Extender criará uma tabela que tem uma coluna espacial, registrará esta coluna como camada e carregará a camada e as outras colunas da tabela com os dados do arquivo.

O sistema de referência espacial especificado no arquivo de transferência SDE será comparado aos sistemas de referência espacial registrados no DB2 Spatial Extender. Se o sistema especificado corresponder a um sistema registrado, os valores negativos e decimais nos dados de transferência serão modificados, quando carregados, do modo estabelecido pelo sistema registrado. Se o sistema especificado não corresponder aos sistemas registrados, o DB2 Spatial Extender criará um novo sistema de referência espacial para estabelecer as modificações.

Autorização

Ao importar dados para uma tabela existente, a ID de usuário com a qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade SYSADM ou DBADM no banco de dados que contém a tabela para a qual os dados serão importados.
- O privilégio CONTROL nesta tabela.

Quando a tabela para qual você deseja importar dados deve ser criada, a ID de usuário com a qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade SYSADM ou DBADM no banco de dados que contém a tabela que deverá ser criada.

Parâmetros de entrada

Tabela 20. Parâmetros de entrada para o procedimento armazenado db2gse.gse_import_sde.

Nome	Tipo de dados	Descrição
layerSchema	VARCHAR(30)	Nome do esquema ao qual pertence a tabela ou view especificados no parâmetro layerTable. Este parâmetro pode ser nulo. **Comentário:** Se você não fornecer um valor para o parâmetro layerSchema, ele assumirá a ID de usuário com que o procedimento armazenado db2gse.gse_import_sde é chamado.
layerTable	VARCHAR(128)	Nome da tabela na qual devem ser carregados os dados de transferência SDE. Este parâmetro não pode ser nulo.
layerColumn	VARCHAR(30)	Nome da coluna que foi registrada como camada na qual serão carregados os dados espaciais do arquivo de transferência SDE. Este parâmetro não pode ser nulo.
fileName	VARCHAR(128)	Nome do arquivo de transferência SDE que deverá ser importado. Este parâmetro não pode ser nulo.
commitScope	INTEGER	Número de registros por ponto de verificação. Este parâmetro pode ser nulo.

Parâmetros de saída

Tabela 21. Parâmetros de saída para o procedimento armazenado db2gse.gse_import_sde.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_import_shape

Use este procedimento armazenado para importar um arquivo de shape para um banco de dados que tenha sido ativado para operações espaciais. O procedimento armazenado pode operar de uma destas duas formas:

- Se o arquivo de shape destina-se a uma tabela existente que tenha uma coluna de camada registrada, o DB2 Spatial Extender carregará a tabela com os dados do arquivo.
- Caso contrário, DB2 Spatial Extender criará uma tabela que tem uma coluna espacial, registrará esta coluna como camada e carregará a camada e as outras colunas da tabela com os dados do arquivo.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseImportShape no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

A ID de usuário com a qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade SYSADM ou DBADM no banco de dados que contém a tabela na qual serão carregados os dados de shape importados.
- O privilégio CONTROL nesta tabela.

Parâmetros de entrada

Tabela 22. Parâmetros de entrada para o procedimento armazenado db2gse.gse_import_shape.

Nome	Tipo de dados	Descrição
layerSchema	VARCHAR(30)	Nome do esquema ao qual pertence a tabela ou view especificados no parâmetro layerTable. Este parâmetro pode ser nulo. **Comentário:** Se você não fornecer um valor para o parâmetro layerSchema, ele assumirá a ID de usuário com que o procedimento armazenado db2gse.gse_import_shape é chamado.
layerTable	VARCHAR(128)	Nome da tabela na qual deve ser carregado o arquivo de shape é importado. Este parâmetro não pode ser nulo.
layerColumn	VARCHAR(30)	Nome da coluna que foi registrada como camada na qual serão carregados os dados de shape. Este parâmetro não pode ser nulo.

Tabela 22. Parâmetros de entrada para o procedimento armazenado db2gse.gse_import_shape. (continuação)

Nome	Tipo de dados	Descrição
fileName	VARCHAR(128)	Nome do arquivo de shape que deverá ser importado. Este parâmetro não pode ser nulo.
exceptionFile	VARCHAR(128)	Caminho e nome do arquivo no qual serão armazenados os shapes que não puderam ser importados. Este é um novo arquivo que será criado quando o procedimento armazenado db2gse.gse_import_shape for executado. Este parâmetro não pode ser nulo.
srId	INTEGER	Identificador do sistema de referência espacial a ser usado pela camada na qual os dados de shape serão carregados. Este parâmetro pode ser nulo. **Comentário:** Se este identificador não estiver especificado, a transformação interna será definida com a resolução máxima possível para o arquivo de shape.
commitScope	INTEGER	Número de registros por ponto de verificação. Este parâmetro pode ser nulo.

Parâmetros de saída

Tabela 23. Parâmetros de saída para o procedimento armazenado db2gse.gse_import_shape.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_register_gc

Use este procedimento armazenado para registrar um geocoder diferente do padrão. Para saber se um geocoder já foi registrado, consulte a view do catálogo DB2GSE.SPATIAL_GEOCODER (descrita em "DB2GSE.SPATIAL_ GEOCODER" na página 114).

Autorização

A ID de usuário com a qual o procedimento armazenado é chamado deve ter autoridade SYSADM ou DBADM no banco de dados que contém o geocoder que este procedimento armazenado registra.

Parâmetros de entrada

Tabela 24. Parâmetros de entrada para o procedimento armazenado db2gse.gse_register_gc.

Nome	Tipo de dados	Descrição
gcId	INTEGER	Identificador numérico do geocoder que você está registrando. Este parâmetro não pode ser nulo. **Comentário:** Este identificador deve ser exclusivo dentro do banco de dados.
gcName	VARCHAR(64)	Descrição breve do geocoder que você está registrando. Este parâmetro não pode ser nulo. **Comentário:** Esta descrição deve ser uma cadeia de caracteres exclusiva dentro do banco de dados.
vendorName	VARCHAR(64)	Nome do fornecedor que ofereceu o geocoder que você está registrando. Este parâmetro não pode ser nulo.
primaryUDF	VARCHAR(256)	Nome completo do geocoder que você está registrando. Este parâmetro não pode ser nulo.
precisionLevel	INTEGER	O grau em que os dados fonte devem coincidir dados de referência correspondentes para que o geocoder processe os dados fonte com êxito. Este parâmetro não pode ser nulo. **Comentário:** O nível de precisão pode variar de 1 a 100%.

Tabela 24. Parâmetros de entrada para o procedimento armazenado db2gse.gse_register_gc. (continuação)

Nome	Tipo de dados	Descrição
vendorSpecific	VARCHAR(256)	Informações técnicas fornecidas pelo fornecedor; por exemplo, o caminho e nome de um arquivo que o fornecedor usa para definir parâmetros. Este parâmetro pode ser nulo.
geoArea	VARCHAR(256)	Área geográfica a ser geocodificada. Este parâmetro pode ser nulo.
descrição	VARCHAR(256)	Observações apresentadas pelo fornecedor Este parâmetro pode ser nulo.

Parâmetros de saída

Tabela 25. Parâmetros de saída para o procedimento armazenado db2gse.gse_register_gc.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_register_layer

Use este procedimento armazenado para registrar uma coluna espacial como camada. Quando este procedimento armazenado estiver processado, as informações sobre a camada que está sendo registrada serão incluídas na view do catálogo DB2GSE.GEOMETRY_COLUMNS. Para obter informações sobre esta view, consulte “DB2GSE.GEOMETRY_COLUMNS” na página 114.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseRegisterLayer no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

A ID de usuário com a qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Para uma camada da tabela:
  - Autoridade SYSADM ou DBADM no banco de dados que contém a tabela à qual pertence esta camada.
  - O privilégio CONTROL ou ALTER nesta tabela.
- Para uma camada da view:
  - O privilégio SELECT na tabela base que contém (1) os dados de endereço que serão geocodificados para esta camada e (2) os dados espaciais que resultarem da geocodificação.

Parâmetros de entrada

Tabela 26. Parâmetros de entrada para o procedimento armazenado db2gse.gse_register_layer.

Nome	Tipo de dados	Descrição
layerSchema	INTEGER(30)	Nome do esquema ao qual pertence a tabela ou view especificados no parâmetro layerTable. Este parâmetro pode ser nulo. **Comentário:** Se você não fornecer um valor para o parâmetro layerSchema, ele assumirá a ID de usuário com que o procedimento armazenado db2gse.gse_register_layer é chamado.
layerTable	VARCHAR(128)	Nome da tabela ou view que contém a coluna que está sendo registrada como camada. Este parâmetro não pode ser nulo.

Tabela 26. Parâmetros de entrada para o procedimento armazenado db2gse.gse_register_layer. (continuação)

Nome	Tipo de dados	Descrição
layerColumn	VARCHAR(128)	Nome da coluna que está sendo registrada como camada. Se a coluna não existir, DB2 Spatial Extender a criará. Este parâmetro não pode ser nulo.
layerTypeName	VARCHAR(64)	Tipo de dados da coluna que está sendo registrada como camada.Você deve especificar o tipo de dados em maiúsculas, por exemplo: `ST_POINT` Este parâmetro não poderá ser nulo se a coluna for uma coluna de tabela que deve ser criada quando esse procedimento armazenado for processado. Caso contrário, se a coluna for uma coluna existente dentro de uma tabela ou view, esse parâmetro poderá ser nulo.
srId	INTEGER	Identificador do sistema de referência espacial usado para esta camada. Este parâmetro não pode ser nulo para a camada de uma tabela. O DB2 Spatial Extender ignora este parâmetro quando uma camada da view é registrada.
geoSchema	VARCHAR(30)	Aplica-se quando uma coluna da view é registrada como camada. O parâmetro geoSchema é o esquema da tabela que sustenta a view à qual pertence a coluna. Este parâmetro pode ser nulo quando você registra a coluna de uma view como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da tabela é registrada como camada. **Comentário:** Se você não fornecer um valor para o parâmetro geoSchema, ele assumirá o valor do parâmetro layerSchema.

Tabela 26. Parâmetros de entrada para o procedimento armazenado db2gse.gse_register_layer. (continuação)

Nome	Tipo de dados	Descrição
geoTable	VARCHAR(128)	Aplica-se quando uma coluna da view é registrada como camada. O parâmetro geoTable é o nome da tabela que sustenta a view à qual pertence a coluna. Este parâmetro pode ser nulo quando você registra a coluna de uma view como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da tabela é registrada como camada.
geoColumn	VARCHAR(128)	Aplica-se quando uma coluna da view é registrada como camada. O parâmetro geoColumn é o nome da coluna da tabela que sustenta esta coluna da view. Este parâmetro pode ser nulo quando você registra a coluna de uma view como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da tabela é registrada como camada.
nAttributes	SMALLINT	Número de colunas que contêm os dados fonte que deverão ser geocodificados para esta camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada.
attr1Name	VARCHAR(128)	Nome da primeira coluna que contém os dados fonte que deverão ser geocodificados para esta camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. Se você pretende usar o geocoder padrão, precisará armazenar os endereços das ruas na coluna attr1Name.

Tabela 26. Parâmetros de entrada para o procedimento armazenado db2gse.gse_register_layer. (continuação)

Nome	Tipo de dados	Descrição
attr2Name	VARCHAR(128)	Nome da segunda coluna que contém os dados fonte que deverão ser geocodificados para essa camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. Se você pretende usar o geocoder padrão, precisará armazenar os nomes das cidades na coluna attr2Name.
attr3Name	VARCHAR(128)	Nome da terceira coluna que contém os dados fonte que devem ser geocodificados para essa camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. Se você pretende usar o geocoder padrão, precisará armazenar os nomes ou abreviações dos estados na coluna attr3Name.
attr4Name	VARCHAR(128)	Nome da quarta coluna que contém os dados fonte que devem ser geocodificados para essa camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. Se você pretende usar o geocoder padrão, precisará armazenar os CEPs na coluna attr4Name.

Tabela 26. Parâmetros de entrada para o procedimento armazenado db2gse.gse_register_layer. (continuação)

Nome	Tipo de dados	Descrição
attr5Name	VARCHAR(128)	Nome da quinta coluna que contém os dados fonte que devem ser geocodificados para essa camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. O gecoder padrão ignora a coluna Attr5Name.
attr6Name	VARCHAR(128)	Nome da sexta coluna que contém os dados fonte que devem ser geocodificados para essa camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. O gecoder padrão ignora a coluna Attr6Name.
attr7Name	VARCHAR(128)	Nome da sétima coluna que contém os dados fonte que devem ser geocodificados para essa coluna. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. O gecoder padrão ignora a coluna Attr7Name.
attr8Name	VARCHAR(128)	Nome da oitava coluna que contém os dados fonte que devem ser geocodificados para essa camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. O gecoder padrão ignora a coluna Attr8Name.

Tabela 26. Parâmetros de entrada para o procedimento armazenado db2gse.gse_register_layer. (continuação)

Nome	Tipo de dados	Descrição
attr9Name	VARCHAR(128)	Nome da nona coluna que contém os dados fonte que devem ser geocodificados para essa camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. O gecoder padrão ignora a coluna Attr9Name.
attr10Name	VARCHAR(128)	Nome da décima coluna que contém os dados fonte que devem ser geocodificados para essa camada. Este parâmetro pode ser nulo quando você registra a coluna de uma tabela como uma camada. O DB2 Spatial Extender ignora este parâmetro quando uma coluna da view é registrada como camada. O gecoder padrão ignora a coluna Attr10Name.

Parâmetros de saída

Tabela 27. Parâmetros de saída para o procedimento armazenado db2gse.gse_register_layer.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

Restrições

- Se você estiver registrando uma coluna da view como camada, este registro deverá estar baseado numa coluna de tabela que já esteja registrada como camada.
- Não mais que dez colunas de atributos podem conter os dados que deverão ser geocodificados para a camada que você está registrando.

db2gse.gse_run_gc

Use este procedimento armazenado para executar um geocoder no modo batch. Para obter informações sobre esta tarefa, consulte “Executando o geocoder no modo batch” na página 43.

Para obter um exemplo do código para chamada deste procedimento armazenado, consulte a função C gseRunGC no programa de amostra. Para obter informações sobre este programa, consulte “Capítulo 8. Gravando aplicações para DB2 Spatial Extender” na página 59.

Autorização

A ID de usuário com a qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade SYSADM ou DBADM no banco de dados que contém a tabela na qual o geocoder especificado deve agir.
- O privilégio CONTROL ou UPDATE nesta tabela.

Parâmetros de entrada

Tabela 28. Parâmetros de entrada para o procedimento armazenado db2gse.gse_run_gc.

Nome	Tipo de dados	Descrição
layerSchema	VARCHAR(30)	Nome do esquema ao qual pertence a tabela ou view especificados no parâmetro layerTable. Este parâmetro pode ser nulo. **Comentário:** Se você não fornecer um valor para o parâmetro layerSchema, ele assumirá a ID de usuário com que db2gse.gse_run_gc será chamado.
layerTable	VARCHAR(128)	Nome da tabela que contém a coluna na qual os dados geocodificados serão inseridos. Este parâmetro não pode ser nulo.
layerColumn	VARCHAR(128)	Nome da coluna na qual os dados geocodificados serão inseridos. Este parâmetro não pode ser nulo.
gcId	INTEGER	Identificador do geocoder que você deseja executar. Este parâmetro pode ser nulo. Para descobrir os identificadores dos geocoders registrados, consulte a view do catálogo DB2GSE.SPATIAL_GEOCODER.

Tabela 28. Parâmetros de entrada para o procedimento armazenado db2gse.gse_run_gc. (continuação)

Nome	Tipo de dados	Descrição
precisionLevel	INTEGER	O grau em que os dados fonte devem coincidir dados de referência correspondentes para que o geocoder processe os dados fonte com êxito. Este parâmetro pode ser nulo. **Comentário:** O nível de precisão pode variar de 1 a 100%.
vendorSpecific	VARCHAR(256)	Informações técnicas fornecidas pelo fornecedor; por exemplo, o caminho e nome de um arquivo que o fornecedor usa para definir parâmetros. Este parâmetro pode ser nulo.
whereClause	VARCHAR(256)	O corpo da cláusula WHERE. Define uma restrição ao conjunto de registros que serão geocodificados. A cláusula pode referenciar qualquer coluna de atributo na tabela em que o geocoder será operado. Este parâmetro pode ser nulo.
commitScope	INTEGER	Número de registros por ponto de verificação. Este parâmetro pode ser nulo.

Parâmetros de saída

Tabela 29. Parâmetros de saída para o procedimento armazenado db2gse.gse_run_gc.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_unregist_gc

Use este procedimento armazenado para cancelar um registro diferente do geocoder padrão.

Para procurar informações sobre o geocoder ao qual você deseja cancelar o registro, consulte a view do catálogo DB2GSE.SPATIAL_GEOCODER, consulte“DB2GSE.SPATIAL_ GEOCODER” na página 114.

Autorização

A ID de usuário com a qual o procedimento armazenado é chamado deve ter autoridade SYSADM ou DBADM no banco de dados que contém o geocoder que deverá ter o registro cancelado.

Parâmetro de entrada

Tabela 30. Parâmetros de entrada para o procedimento armazenado db2gse.gse_unregist_gc.

Nome	Tipo de dados	Descrição
gcId	INTEGER	O identificador do geocoder que deverá ter o registro cancelado. Este parâmetro não pode ser nulo.

Parâmetros de saída

Tabela 31. Parâmetros de saída para o procedimento armazenado db2gse.gse_unregist_gc.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

db2gse.gse_unregist_layer

Use este procedimento armazenado para cancelar o registro de uma camada. O procedimento armazenado faz isto:

- Removendo a definição da camada das tabelas de catálogo do DB2 Spatial Extender.
- Excluindo a restrição de verificação que o DB2 Spatial Extender colocou na tabela base desta camada para assegurar que os dados espaciais da camada atendam aos requisitos do sistema de referência espacial da camada.
- Eliminando os disparadores que são usados para atualizar a coluna espacial sempre que os dados do endereço forem incluídos, alterados ou removidos.

Quando este procedimento armazenado estiver processado, as informações sobre a camada serão removidas da metaview DB2GSE.GEOMETRY_COLUMNS. Para obter informações sobre esta view, consulte “DB2GSE.GEOMETRY_COLUMNS” na página 114.

Autorização

A ID de usuário com a qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Para uma camada da tabela:
  - Autoridade SYSADM ou DBADM no banco de dados que contém a tabela base desta camada.
  - O privilégio CONTROL ou ALTER nesta tabela.
- Para uma camada da view:
  - O privilégio SELECT na tabela base que contém (1) os dados de endereço que serão geocodificados para esta camada e (2) os dados espaciais que resultarem da geocodificação.

Parâmetros de entrada

Tabela 32. Parâmetros de entrada para o procedimento armazenado db2gse.gse_unregist_layer.

Nome	Tipo de dados	Descrição
layerSchema	VARCHAR(30)	Nome do esquema ao qual pertence a tabela especificada no parâmetro layerTable. Este parâmetro pode ser nulo. **Comentário:** Se você não fornecer um valor para o parâmetro layerSchema, ele assumirá a ID de usuário com que o procedimento armazenado db2gse.gse_unregister_layer é chamado.

Tabela 32. Parâmetros de entrada para o procedimento armazenado db2gse.gse_unregist_layer. (continuação)

Nome	Tipo de dados	Descrição
layerTable	VARCHAR(128)	Nome da tabela que contém a coluna especificada no parâmetro layerColumn. Este parâmetro não pode ser nulo.
layerColumn	VARCHAR(128)	Nome da coluna espacial que foi definida como a camada que você deseja cancelar o registro. Este parâmetro não pode ser nulo.

Parâmetros de saída

Tabela 33. Parâmetros de saída para o procedimento armazenado db2gse.gse_unregist_layer.

Nome	Tipo de dados	Descrição
msgCode	INTEGER	Código associado às mensagens que o responsável pela chamada deste procedimento armazenado poderá retornar.
Reservado	VARCHAR(1024)	Mensagem de erro completa, conforme construída no servidor DB2 Spatial Extender.

Restrição

Se uma coluna da view definida como camada da view estiver baseada numa coluna da tabela, que foi definida como uma camada da tabela, não será possível cancelar o registro desta camada da tabela até que o registro da camada da view seja cancelado.

Capítulo 10. Mensagens

Este capítulo documenta mensagens que o DB2 Spatial Extender retorna aos usuários. Cada mensagem possui um identificador. Identificadores que terminam com a letra E são para as mensagens de erro; aquelas que terminam com W são para avisos; e aquelas que terminam com I são para informações gerais.

DBA7200E Mais de 10 colunas estão selecionadas como entrada para um geocoder

Explicação: Até 10 colunas podem ser selecionadas como entrada para um geocoder

Resposta do Usuário: Mova os nomes das colunas da caixa colunas Selecionadas para a caixa colunas Disponíveis até que a caixa Colunas Selecionadas liste dez nomes ou menos.

DBA7201E O banco de dados não está ativado para operações do Spatial Extender.

Explicação: O banco de dados deve estar ativo para o Spatial Extender antes que você possa utilizá-lo.

Resposta do Usuário: Dê um clique com o botão direito no banco de dados e selecione Spatial Extender —> Ativar a partir dos menus.

GSE0000I A operação foi concluída com sucesso.

GSE0001E O Spatial Extender não pôde executar a operação solicitada (“<nome da operação>”) sob a ID de usuário “<id do usuário>”.

Explicação: Você solicitou esta operação sob uma ID de usuário que não mantém o privilégio ou autoridade para executar a operação.

Resposta do Usuário: Consulte a documentação para localizar a autorização apropriada ou obtê-la de um administrador do Spatial Extender.

GSE0002E O “<valor>” não é um valor válido para o argumento “<nome do argumento>”.

Explicação: O valor que você digitou estava incorreto.

Resposta do Usuário: Consulte a documentação ou um administrador do Spatial Extender para localizar qual valor ou faixa de valores você precisa especificar.

GSE0003E O Spatial Extender não pôde executar a operação solicitada porque o argumento “<nome do argumento>” não foi especificado.

Explicação: Você não especificou um argumento que é requerido para esta operação.

Resposta do Usuário: Especifique o argumento “<nome do argumento>” com o valor que deseja; em seguida solicite a operação novamente.

GSE0004W O argumento “<nome do argumento>” não foi avaliado.

Explicação: A operação que você solicitou não usa o argumento “<nome do argumento>”.

Resposta do Usuário: Nenhuma ação é necessária.

GSE0005E O Spatial Extender não pôde processar sua solicitação para criar um objeto chamado “<nome do objeto>”.

Explicação: Ou o objeto “<nome do objeto>” já existe, ou você não possui a permissão apropriada para criá-lo. Ele pode ser uma tabela, coluna, disparador, índice, arquivo ou outro tipo de objeto.

Resposta do Usuário: Se o “<nome do objeto>” é o objeto que você deseja, não faça nada. Caso contrário, especifique o nome corretamente e verifique se você tem a permissão correta para criar o objeto.

GSE0006E O Spatial Extender não pôde executar a operação solicitada em um objeto ativo ou registrado chamado “<nome do objeto>”.

Explicação: O objeto “<nome do objeto>” já ativo ou registrado, ou ele já existe. Ele pode ser uma camada, índice, sistema de referência espacial, sistema de coordenadas, geocoder, ou outro tipo de objeto.

Resposta do Usuário: Tenha a certeza de que o objeto “<nome do objeto>” existe e submeta novamente a sua solicitação.

GSE0007E O Spatial Extender não pôde executar a operação solicitada no “<nome do objeto>”, um objeto que ainda não foi ativado ou registrado.

Explicação: O objeto “<nome do objeto>” não foi ativado ou registrado. Ele pode ser uma camada, índice, sistema de referência espacial, sistema de coordenada espacial, geocoder, ou outro tipo de objeto.

Resposta do Usuário: Ative ou registre o objeto “<nome do objeto>”. Em seguida, submeta novamente a sua solicitação.

GSE0008E Ocorreu um erro SQL inesperado (“<mensagem de erro sql>”).

Resposta do Usuário: Procure a mensagem detalhada associada com o SQLCODE na mensagem de erro SQL “<mensagem de erro sql>”. Se necessário, entre em contato com o representante de serviços IBM.

GSE0009E A operação solicitada não pôde ser executada em um objeto chamado “<nome do objeto>” que já existe.

Explicação: O “<nome do objeto>” já existe no banco de dados ou no sistema operacional. Ele pode ser um arquivo, tabela, view, coluna, índice, disparador ou outro tipo de objeto.

Resposta do Usuário: Assegure-se de que tenha especificado o objeto corretamente ao tentar acessá-lo. Se necessário, exclua o objeto.

GSE0010E A operação solicitada não pôde ser executada em um objeto chamado “<nome do objeto>” que pode não existir.

Explicação: O “<nome do objeto>” não existe no banco de dados ou no sistema operacional. Ele pode ser um arquivo, tabela, view, coluna, índice, disparador ou outro tipo de objeto.

Resposta do Usuário: Assegure-se de ter a permissão correta para acessar o objeto. Caso tenha esta permissão e o objeto não existe, então você precisa criá-lo.

GSE0011E O Spatial Extender não pôde desativar ou desregistrar o objeto “<nome do objeto>”.

Explicação: O “<nome do objeto>” é dependente de outro objeto. O “<nome do objeto>” pode ser um sistema de referência espacial, camada, geocoder ou outro tipo de objeto.

Resposta do Usuário: Consulte a documentação para localizar quais tipos de objetos “<nome do objeto>” podem ser dependentes. Em seguida

remova o objeto específico “<nome do objeto>” que é dependente.

GSE0012E O Spatial Extender não pôde processar sua solicitação porque a coluna espacial completamente qualificada “<esquema da camada.nome da camada.coluna da camada>” não está registrada como uma camada de tabela.

Explicação: A coluna espacial completamente qualificada “<esquema da camada.nome da camada.coluna da camada>” deve estar registrada como uma camada de tabela antes que você possa executar certas operações associadas com a mesma (por exemplo, ativar seu índice, ativar um geocoder para preenchê-la no modo batch ou para atualizá-la automaticamente).

Resposta do Usuário: Assegure-se de que a coluna espacial completamente qualificada “<esquema da camada.nome da camada.coluna da camada>” esteja registrada como camada de tabela verificando a view DB2GSE.GEOMETRY_COLUMNS no catálogo do Spatial Extender. Também esteja certo de que a tabela que contém esta coluna também inclui colunas de atributo correspondente válido.

GSE0013E O banco de dados não está ativados para operações espaciais.

Explicação: O banco de dados não está ativado para operações espaciais. Portanto, o catálogo do Spatial Extender não existe.

Resposta do Usuário: Ativar o banco de dados para operações espaciais.

GSE0014E O banco de dados foi ativado para operações espaciais.

Explicação: O banco de dados foi ativado para operações espaciais.

Resposta do Usuário: Verificar se o banco de dados foi ativado como esperado. Se necessário, desabilite o banco de dados.

GSE0498E Ocorreu o seguinte erro: “<mensagem de erro>”.

GSE0499W O Spatial Extender emitiu o seguinte aviso: “<mensagem de erro>”.

GSE0500E O modo de operação que você especificou (“<modo de operação>”) não é válido.

Explicação: O modo especificado não é suportado pela operação que você solicitou.

Resposta do Usuário: Consulte a documentação para localizar quais modos são suportados pela operação.

GSE1001E O Spatial Extender foi incapaz de registrar uma camada de view que é chamada “<nome do esquema.nome da view.nome da coluna>” e que está baseada na coluna espacial “<nome do esquema.nome da tabela.nome da coluna>”.

Explicação: A coluna espacial que você especificou (“<nome do esquema.nome da tabela.nome da coluna>”) não foi registrada como uma camada de tabela.

Resposta do Usuário: Registre a coluna “<nome do esquema.nome da tabela.nome da coluna>” como uma camada de tabela.

GSE1002E O Spatial Extender foi incapaz de registrar uma camada de view que é chamada “<nome do esquema.nome da view.nome da coluna>” e que está baseada na tabela “<nome do esquema.nome da tabela>”.

Explicação: A tabela que você especificou (“<nome do esquema.nome da tabela>”) não suporta a view “<nome do esquema.nome da view.nome da coluna>”, direta ou indiretamente.

Resposta do Usuário: Localize qual tabela base

é para a view “<nome do esquema.nome da view.nome da coluna>”, e especifique-a.

GSE1003E **O Spatial Extender foi incapaz de acessar uma coluna chamada “<nome da coluna>” em uma tabela ou view chamada “<nome do esquema.nome do objeto>”.**

Explicação: A tabela ou view “<nome do esquema.nome do objeto>” não possui uma coluna chamada “<nome da coluna>”.

Resposta do Usuário: Verifique a definição da tabela ou view “<nome do esquema.nome do objeto>” para localizar o nome próprio da coluna que você deseja.

GSE1004E **O Spatial Extender foi incapaz de registrar a coluna espacial completamente qualificada “<nome do esquema.nome da tabela.nome da coluna>” como uma camada de tabela.**

Explicação: A coluna “<nome do esquema.nome da tabela.nome da coluna>”não possui um tipo de dados espacial, ou não está associada a uma tabela base.

Resposta do Usuário: Defina um tipo de dados espacial para a coluna “<nome do esquema.nome da tabela.nome da coluna>”, ou assegure-se de que esta coluna seja parte de uma tabela base local.

GSE1005E **O sistema de referência espacial (“<id de referência espacial da camada da view>”) que você especificou para uma camada de view difere do sistema de referência espacial (“<id de referência espacial da camada da view>”) que é utilizado para esta camada de tabela de suporte de camadas.**

Explicação: Um sistema de referência espacial de camadas de view deve ser o mesmo que o sistema de referência espacial de camadas de tabela de suporte.

Resposta do Usuário: Especifique o sistema de referência espacial da camada de tabela de suporte para a camada da view.

GSE1006E **Porque a “<id de referência espacial>” é uma ID de sistema de referência espacial, o Spatial Extender foi incapaz de registrar a camada que você solicitou.**

Explicação: O sistema de referência espacial que você especificou (“<id de referência espacial>”) não foi ativado ou registrado.

Resposta do Usuário: Ativar ou registrar o sistema de referência espacial. Então, submeta novamente sua solicitação para registrar a camada.

GSE1007E **Um erro SQL (SQLSTATE “<sqlstate>”) pode ter ocorrido quando o Spatial Extender tentou sem sucesso incluir uma coluna espacial (“<nome da coluna>”) para a tabela “<nome do esquema.nome da tabela>”.**

Resposta do Usuário: Procure a mensagem associada com o SQLSTATE “<sqlstate>”.

GSE1008E **O DB2 Spatial Extender foi incapaz de registrar uma camada de view “<esquema da camada.nome da camada.coluna da camada>” porque o tipo de dados espacial “<tipo de coluna da camada>” da camada da view não coincide com o tipo de dados espacial “<tipo de coluna geométrica>” da camada de tabela suportada “<esquema de geometria.nome da geometria.coluna de geometria>”.**

Explicação: O tipo de dados espacial de uma camada de view “<esquema da camada.nome da camada.coluna da camada>” deve coincidir com o tipo de dados espacial da camada de tabela que suporta a camada “<esquema de geometria.nome de geometria.coluna de

geometria>". A inconsistência entre estes dois tipos de dados causa ambigüidade quando os dados espaciais são processados.

Resposta do Usuário: Assegure-se de que os tipos de dados espaciais da camada de view e suas camadas de tabelas suportadas sejam as mesmas.

GSE1020E **A "<'id de referência espacial>" é uma ID de sistema de referência espacial inválida.**

Explicação: Um sistema de referência espacial com um identificador de "<'id de referência espacial>" não foi ativado.

Resposta do Usuário: Assegure-se de que a referência espacial especificada foi ativada.

GSE1021E **O Spatial Extender não pôde ativar o sistema de referência espacial "<id de referência espacial>" porque a correspondente ID do sistema de coordenada espacial "<id de coordenada espacial>" era inválida.**

Explicação: Um sistema de coordenada com um identificador de "<id de coordenada espacial>" não está definido no catálogo Spatial Extender.

Resposta do Usuário: Verifique o identificador do sistema de coordenada "<id de coordenada espacial>" conferindo a view DB2GSE.COORD_REF_SYS no catálogo do Spatial Extender.

GSE1030E **Porque a "<nome do esquema.nome da tabela>" não é uma tabela base, o Spatial Extender não pôde ativar um geocoder para a mesma.**

Explicação: O objeto que contém os dados fonte que você deseja geocodificar deve ser uma tabela base.

Resposta do Usuário: Esteja seguro de que as colunas que contém os dados fonte que você deseja geocodificar sejam parte de uma tabela base.

GSE1031E **O Spatial Extender não pôde ativar o geocoder "<id do geocoder>" para operar automaticamente no modo criar para a camada "<esquema da camada.nome da camada.coluna da camada>".**

Explicação: As possíveis explicações são:

- O geocoder já está ativo para atualizar a camada "<esquema da camada.nome da camada.coluna da camada>" automaticamente.
- O geocoder foi temporariamente invalidado para esta canada.
- Nenhuma coluna para dados fonte foi definida para esta camada.

Resposta do Usuário: Se o geocoder foi temporariamente invalidado, ative-o para operar automaticamente no modo ″Recriar″.

GSE1032E **O Spatial Extender não pôde ativar o geocoder "<id do geocoder>" para operar automaticamente no modo re-criar para a camada "<esquema da camada.nome da camada.coluna da camada>".**

Explicação: As possíveis explicações são:

- O geocoder já está ativo para atualizar a camada "<esquema da camada.nome da camada.coluna da camada>" automaticamente.
- O geocoder não foi invalidado previamente para esta camada.
- Nenhuma coluna para dados fonte foi definida para esta camada.

Resposta do Usuário: Se o geocoder foi desativado previamente no modo eliminar, ou se ele nunca foi definido para esta camada, ative-o para operar automaticamente no modo ″Criar″.

GSE1033E Ocorreu um erro SQL quando o Spatial Extender tentou incluir disparadores para uma tabela que contém a coluna para a camada "<esquema da camada.nome da camada.coluna da camada>" (SQLSTATE "<sqlstate>").

Explicação: O propósito dos disparadores é manter a integridade dos dados entre as colunas de atributos de onde sucede a entrada do geocoder e a coluna espacial que suas saídas vão. O erro SQL ocorrido quando o DB2 tentou criar estes disparadores.

Resposta do Usuário: Procure a mensagem associada com o SQLSTATE "<sqlstate>".

GSE1034E O Spatial Extender não pôde desativar o geocoder "<id do geocoder>" no modo eliminar para a camada "<esquema da camada.nome da camada.coluna da camada>".

Explicação: As possíveis explicações são:

- O geocoder nunca foi ativado para atualizar a camada "<esquema da camada.nome da camada.coluna da camada>" automaticamente.
- O geocoder foi desativado no modo eliminar.

Resposta do Usuário: Determine o estado do geocoder antes de tentar desativá-lo. Por exemplo, ele foi registrado? foi ativado? Então decida se ele precisa ser desativado no modo eliminar. Por exemplo, caso nunca tenha sido ativado, talvez não seja preciso desativá-lo por completo.

GSE1035E O Spatial Extender não pôde desativar o geocoder "<id do geocoder>" no modo invalidar para a camada "<esquema da camada.nome da camada.coluna da camada>".

Explicação: As possíveis explicações são:

- O geocoder nunca foi ativado para atualizar a camada "<esquema da camada.nome da camada.coluna da camada>" automaticamente.
- O geocoder foi desativado no modo invalidado ou no modo eliminar.

Resposta do Usuário: Determine o estado do geocoder antes de tentar desativá-lo. Por exemplo, ele foi registrado? Foi ativado? Então decida se ele precisa ser desativado no modo invalidar. Por exemplo, caso já tenha sido desativado no modo invalidar, talvez não seja preciso desativá-lo neste modo uma segunda vez.

GSE1036E Ocorreu um erro SQL quando o Spatial Extender tentou eliminar disparadores de uma tabela que contém a coluna para a camada "<esquema da camada.nome d acamada.coluna da camada>" (SQLSTATE "<sqlstate>").

Explicação: Os disparadores foram criados para manter a integridade dos dados entre as colunas de atributos de onde sucede a entrada do geocoder e a coluna espacial para onde as suas saídas vão. O erro SQL ocorrido quando o DB2 tentou eliminar estes disparadores.

Resposta do Usuário: Procure a mensagem associada com o SQLSTATE "<sqlstate>".

GSE1037E O Spatial Extender não pôde fazer o geocode dos dados fonte para a camada de tabela "<esquema da camada.nome da camada.coluna da camada>", possivelmente porque um valor incorreto "<número de atributos>" foi assinalado para o argumento que especifica quantas são as colunas de atributo para fornecer dados fonte para esta camada/

Explicação: O número de colunas de atributo associadas a esta camada foi especificado incorretamente, ou o nome de uma ou mais destas colunas foi especificado incorretamente.

Resposta do Usuário: Assegure-se de que esta camada esteja registrada com o número correto e com os nomes das colunas de atributo associadas, ou verifique falta de exatidão dos dados de entrada e saída para o geocoder.

GSE1038E Ocorreu um erro SQL quando o Spatial Extender tentou fazer o geocode dos dados fonte para a camada de tabela "<esquema da camada.nome da camada.coluna da camada>" no modo batch (SQLSTATE "<sqlstate>").

Resposta do Usuário:

- Procure a mensagem associada com o SQLSTATE "<sqlstate>".
- Assegure-se de que o conteúdo e o argumento primaryUDF desta camada estejam definidos corretamente.

GSE1050E O tamanho da grade que você especificou "<tamanho da grade>") é inválido para o primeiro nível da grade.

Explicação: Você especificou zero ou um número negativo como tamanho da grade para o primeiro nível da grade.

Resposta do Usuário: Especificar um número positivo como tamanho da grade.

GSE1051E O tamanho da grade que você especificou "<tamanho da grade>") é inválido para o segundo e terceiro níveis da grade.

Explicação: Você especificou um número negativo como tamanho da grade para o segundo ou o terceiro nível de grade.

Resposta do Usuário: Especificar zero ou um número positivo como o tamanho da grade.

GSE1052E Ocorreu um erro SQL quando o Spatial Extender tentou criar o índice espacial "<esquema do índice.coluna do índice>" para uma camada de tabela "<esquema da camada.nome da camada.coluna da camada>" (SQLSTATE "<sqlstate>").

Resposta do Usuário:

- Assegure-se de que o índice espacial esteja especificado corretamente e que a coluna espacial não possui índice associado.
- Procure a mensagem que é associada ao SQLSTATE "<sqlstate>".

GSE1500I O registro fonte "<número do registro>" foi geocodificado com sucesso.

Explicação: Um registro contendo dados de atributo foi geocodificado com sucesso.

GSE1501W O registro fonte "<número do registro>" não foi geocodificado.

Explicação: O nível de precisão foi muito alto.

Resposta do Usuário: Faça o geocode com um nível de precisão menor.

GSE1502W O registro fonte "<número do registro>" não foi encontrado.

Resposta do Usuário: Determine se o registro existe no banco de dados.

GSE2001E O arquivo de transferência especificado ("<nome do arquivo>") não é válido.

Resposta do Usuário: Verifique se o arquivo especificado é um arquivo de transferência SDE, e se o nome do caminho está apropriadamente especificado.

GSE2002E A cláusula fornecida SQL WHERE ("<cláusula SQL where>") não é válida.

Resposta do Usuário: Verifique a sintaxe SQL apropriada para a cláusula WHERE, erros de ortografia e nomes de colunas inválidos.

GSE2003E O valor do shape fornecido não é oficial.

Resposta do Usuário: Verifique para assegurar-se de que o shape fornecido coincide com o tipo especificado da coluna espacial. Se os

tipos coincidem, ou são compatíveis, então a forma da geometria é ilegal. Verifique a sobreposição de polígonos, arcos de ponto único, etc.

GSE2004E O esquema de transferência de arquivo é incompatível com o esquema da camada especificada.

Resposta do Usuário: Verifique para assegurar-se de que os nomes de esquema e camada estejam especificados apropriadamente. Se os esquemas não coincidem, carregue os dados como uma nova tabela e resolva as diferenças do esquema.

GSE2005E O tipo de geometria do arquivo de transferência é incompatível com o tipo de geometria da camada especificada.

Resposta do Usuário: Verifique para assegurar-se de que os nomes de esquema e camada estejam especificados apropriadamente.

GSE2006E Ocorreu um erro de E/S para um arquivo chamado "<nome do arquivo>".

Resposta do Usuário: Verifique se o arquivo existe, se você possui o acesso apropriado para o arquivo e se o arquivo não está sendo utilizado por um outro usuário.

GSE2007E Ocorreu um erro no atributo de conversão.

Resposta do Usuário: Verifique para estar seguro de que todos os tipos de atributo na tabela são suportados - por exemplo, dados BLOB não são suportados nos arquivos de shape Verifique também valores de dados fora da faixa ou valores de dados ilegais tais como datas ruins.

GSE2008E A função importar/exportar foi executada sem memória.

Resposta do Usuário: Verifique para adequar a disponibilidade de memória.

Capítulo 11. Views do catálogo

As views do catálogo do DB2 Spatial Extender contêm metadados em:

- Sistemas coordenados que você pode utilizar. Para obter mais informações, tais como anotações de texto e identificadores destes sistemas, consulte "DB2GSE.COORD_REF_SYS".
- Colunas espaciais que foram registradas como camadas. Para obter mais informações, tais como sistemas de referência espacial associados, tipos de dados e nomes destas colunas, consulte "DB2GSE.GEOMETRY_COLUMNS" na página 114.
- Geocoders que você pode utilizar. Para informações tais como identificadores do geocoder e as regiões que contêm as localizações que os geocoders processam, consulte "DB2GSE.SPATIAL_ GEOCODER" na página 114.
- Sistemas de referência espacial que podem ser usados. Para informações, tais como seus identificadores e descrições dos mesmos, consulte "DB2GSE.SPATIAL_ REF_SYS" na página 115.

DB2GSE.COORD_REF_SYS

Ao ativar um banco de dados para operações espaciais, o DB2 Spatial Extender registra os sistemas coordenados que você pode usar em uma tabela de catálogo. As colunas selecionadas desta tabela compreendem a view de catálogo DB2GSE.COORD_REF_SYS, que é descrita na Tabela 34.

Tabela 34. Colunas na view de catálogo DB2GSE.COORD_REF_SYS

Nome	Tipo de Dados	Permite nulos?	Conteúdo
SCID	INTEGER	Não	Identificador numérico inteiro para este sistema coordenado.
SC_NAME	VARCHAR(64)	Não	Nome deste sistema coordenado.
AUTH_NAME	VARCHAR(256)	Sim	Nome da organização que compilou o sistema coordenado agrega à; por exemplo, o European Petroleum Survey Group (EPSG).
AUTH_SRID	INTEGER	Sim	Um identificador numérico designado para este sistema coordenado, pela organização especificada na coluna AUTH_NAME column.
DESC	VARCHAR(256)	Sim	Descrição deste sistema coordenado.

Tabela 34. Colunas na view de catálogo DB2GSE.COORD_REF_SYS (continuação)

Nome	Tipo de Dados	Permite nulos?	Conteúdo
SRTEXT	VARCHAR(2048)	Não	Texto de anotação para este sistema coordenado.

DB2GSE.GEOMETRY_COLUMNS

Ao criar uma camada, o DB2 Spatial Extender registra-a gravando seu identificador e a informação relacionada à mesma em uma tabela de catálogo. As colunas selecionadas desta tabela compreendem a view de catálogo DB2GSE.GEOMETRY_COLUMNS, que é descrita na Tabela 35.

Tabela 35. Colunas na view de catálogo DB2GSE.GEOMETRY_COLUMNS

Nome	Tipo de Dados	Permite nulos?	Conteúdo
LAYER_CATALOG	VARCHAR(30)	Sim	Nome completo desta camada.
LAYER_SCHEMA	VARCHAR(30)	Não	Esquema da tabela ou view que contém a coluna que foi registrada como esta camada.
LAYER_NAME	VARCHAR(128)	Não	Nome da tabela ou view que contém a coluna que foi registrada como esta camada.
LAYER_COLUMN	VARCHAR(30)	Não	Nome da coluna que foi registrada como esta camada.
GEOMETRY_TYPE	INTEGER	Não	Tipo de dados da coluna que foi registrada como esta camada.
SRID	INTEGER	Não	Identificador do sistema de referência espacial usado para os valores na coluna que foi registrada como esta camada.
STORAGE_TYPE	INTEGER	Sim	Informação de como o DB2 armazena os valores na coluna que foi registrada nesta camada. Por exemplo, dados em STORAGE_TYPE podem indicar que os valores são armazenados como objetos grandes (LOBs) ou como instâncias de tipos de dados abstratos (ADTs).

DB2GSE.SPATIAL_ GEOCODER

Os geocoders disponíveis estão registrados em uma tabela do catálogo. As colunas selecionadas desta tabela compreendem a view de catálogo DB2GSE.SPATIAL_GEOCODER, que é descrita na Tabela 36 na página 115.

Tabela 36. Colunas na view de catálogo DB2GSE.SPATIAL_GEOCODER

Nome	Tipo de Dados	Permite nulos?	Conteúdo
GCID	INTEGER	Não	Identificador numérico deste geocoder.
GC_NAME	VARCHAR(64)	Não	Pequena descrição deste geocoder.
VENDOR_NAME	VARCHAR(128)	Não	Nome do fornecedor que forneceu este geocoder.
PRIMARY_UDF	VARCHAR(256)	Não	Nome completo deste geocoder.
PRECISION_LEVEL	INTEGER	Não	O grau de coincidência dos dados fonte com os dados de referência correspondente a fim de ser processado com sucesso pelo geocoder.
VENDOR_SPECIFIC	VARCHAR(256)	Sim	Caminho para, e nome de, um arquivo que um fornecedor pode utilizar para definir quaisquer parâmetros especiais que este geocoder suporta.
GEO_AREA	VARCHAR(256)	Sim	Área geográfica que contém as localizações a serem geocodificadas.
DESCRIPTION	VARCHAR(256)	Sim	Observações fornecidas pelo fornecedor.

DB2GSE.SPATIAL_ REF_SYS

Ao criar um sistema de referência espacial, o DB2 Spatial Extender registra-a gravando seu identificador e a informação relacionada à mesma em uma tabela de catálogo. As colunas selecionadas desta tabela compreendem a view de catálogo DB2GSE.SPATIAL_REF_SYS, que é descrita na Tabela 37.

Tabela 37. Colunas na view do catálogo DB2GSE.SPATIAL_REF_SYS

Nome	Tipo de Dados	Permite nulos?	Conteúdo
SRID	INTEGER	Não	Identificador definido pelo usuário para este sistema de referência espacial.
SR_NAME	VARCHAR(64)	Não	Nome deste sistema de referência espacial.
SCID	INTEGER	Não	Identificador numérico para o sistema coordenado que suporta este sistema de referência espacial.
SC_NAME	VARCHAR(64)	Não	Nome do sistema coordenado que suporta este sistema de referência espacial.
AUTH_NAME	VARCHAR(256)	Sim	Nome da organização que define os padrões para este sistema de referência espacial.

Tabela 37. Colunas na view do catálogo DB2GSE.SPATIAL_REF_SYS (continuação)

Nome	Tipo de Dados	Permite nulos?	Conteúdo
AUTH_SRID	INTEGER	Sim	O identificador que a organização especificada na coluna AUTH_NAME assinala para este sistema de referência espacial.
SRTEXT	VARCHAR(2048)	Não	Texto de anotação para este sistema de referência espacial.

Capítulo 12. Índices espaciais

Como as colunas espaciais contêm dados geográficos bidimensionais, as aplicações que consultam estas colunas precisam de uma estratégia de índice que identifique rapidamente todas as geometrias contidas em uma determinada extensão. Por este motivo, o DB2 Spatial Extender fornece o índice espacial com três fileiras baseado em uma grade.

Este capítulo descreve este tipo de índice e fornece diretrizes sobre o uso do mesmo. Os tópicos abrangidos são:

Fragmento de um programa de amostra

Considere o seguinte exemplo de como um índice é criado e utilizado no SQL. Você pode consultar a *SQL Reference* para obter maiores informações sobre os comandos CREATE INDEX e CREATE INDEX EXTENSION. Observe que depois que o índice é criado, você pode emitir instruções DDL e DML padrões que utilizam as funções e predicados espaciais.

```
create table customers (cid int, addr varchar(40), ..., loc db2gse.ST_Point)
create table stores (sid int, addr varchar(40), ..., loc db2gse.ST_Point,
 zone db2gse.ST_Polygon)

create index customersx1 on customers(loc) extend using spatial_index(10e0,
 100e0, 1000e0)
create index storesx1 on stores(loc) extend using spatial_index(10e0, 100e0,
 1000e0)
create index storesx2 on stores(zone) extend using spatial_index(10e0, 100e0,
 1000e0)

insert into customers (cid, addr, loc) values
(:cid, :addr, sdeFromBinary(:loc))
insert into customers (cid, addr, loc) values
(:cid, :addr, geocode(:addr))
insert into stores (sid, addr, loc) values
(:sid, :addr, sdeFromBinary(:loc))

update stores set zone = db2gse.ST_Buffer (loc, 2)

select cid, loc from customers
 where db2gse.ST_Within(loc, :polygon) = 1
```

```
select cid, loc from customers
 where db2gse.ST_Within(loc, :circle1) = 1 OR
       db2gse.ST_Within(loc, :circle2) = 1

select c.cid, loc from customers c, stores s
 where db2gse.ST_Contains(s.zone, c.loc) = 1 selectivity 0.01

select avg(c.income) from customers c
 where not exist (select * from stores s
                  where db2gse.ST_Distance(c.loc, s.loc) < 10)
```

Índices B tree

A tecnologia de indexação espacial baseia-se no índice B tree hierárquico tradicional, mas é significativamente diferente. O índice espacial utiliza a *indexação de grade* que é projetada para indexar colunas espaciais bidimensionais. O índice B tree pode manipular apenas dados mono-dimensionais e não pode ser utilizado com informações GIS. Esta seção descreve como um índice B tree é estruturado e utilizado.

O nível mais elevado de um índice B tree, chamado de nó raiz, contém uma chave para cada nó no nível seguinte. O valor de cada chave é o maior valor de chave existente para o nó correspondente no nível seguinte. Dependendo do número de valores na tabela base, é possível que sejam necessários diversos nós intermediários. Estes nós formam uma ponte entre o nó raiz e os nós folha que retém as efetivas IDs de linha da tabela base.

O gerenciador de banco de dados faz pesquisas em um índice B tree começando no nó raiz. Depois, ele continua pelos nós intermediários até chegar no nó folha com a ID de linha da tabela base.

O índice B tree não pode ser aplicado em uma coluna espacial pois a característica bidimensional da coluna espacial exige a estrutura de um índice espacial. Pelo mesmo motivo, não é possível aplicar um índice espacial em uma coluna não-espacial. Além disso, um índice espacial não pode ser aplicado em uma coluna composta de qualquer tipo.

Formas de se criar um índice espacial

Há várias formas de se criar um índice espacial:

- Definir um a partir da janela Criar Índice Espacial. Para obter instruções, consulte “Capítulo 6. Criando índices espaciais” na página 53.

- Solicitar o procedimento armazenado db2gse.gse_enable_idx em um programa da aplicação. Para obter informações sobre este procedimento armazenado, consulte “Capítulo 9. Procedimentos armazenados” na página 69.
- Emitir o comando **db2 create index** com a função **spatial_index** na cláusula USING. Por exemplo:

  ```
  create index storesx1 on customers (loc) using spatial_index(10e0,
  100e0, 1000e0)
  ```

A natureza dos dados espaciais exige que o projetista de banco de dados compreenda sua distribuição relativa de tamanho. O projetista deve determinar o tamanho satisfatório e o número de níveis de grade com os quais criar o índice espacial.

Os níveis de grade, <nível de grade 1>, <nível de grade 2> e <nível de grade 3>, são fornecidos aumentando-se o tamanho da célula. Conseqüentemente, o segundo nível deve ter um tamanho de célula maior do que o primeiro e o terceiro maior que o segundo. O primeiro nível de grade é obrigatório, mas você pode desativar o segundo e o terceiro com um valor zero de precisão dupla (0.0e0).

Como é gerado um índice espacial

Um índice espacial é gerado com o uso de *envelopes*. O envelope é uma geometria em si e representa a extensão mínima e máxima de X e Y de uma geometria. Para maioria das geometrias, o envelope é uma caixa, mas para linhas horizontais e verticais, o envelope é uma linha com dois pontos. Para os pontos, o envelope é o próprio ponto. Para obter mais informações sobre envelopes, consulte “Envelope” na página 131.

O índice espacial é construído em uma coluna espacial produzindo uma ou mais entradas para as interseções de cada envelope da geometria com a grade. Uma interseção é registrada como a ID interna da geometria e das coordenadas mínimas X e Y da célula de grade atravessada. Por exemplo, o polígono na Figura 7 na página 120 intersecta a grade nas coordenadas (20,30), (30,30), (40,30), (20,40), (30,40), (40,40), (20,50), (30,50) e (40,50). Consulte Tabela 38 na página 120 para obter as coordenadas mínimas X e Y para todas as geometrias na Figura 7 na página 120.

Se houver vários níveis de grade, o DB2 Spatial Extender tenta utilizar o nível de grade mais baixo possível. Quando uma geometria intersecta quatro ou mais células da grade no nível fornecido, ela é promovida para o próximo nível mais alto. Sendo assim, um índice espacial que possui três níveis de grade de 10.0e0, 100.0e0 e 1000.0e0, o DB2 Spatial Extender intersecta primeiro cada geometria com a grade de nível 10.0e0. Se uma geometria intersecta quatro ou mais células de grade 10.0e0, ela é promovida e faz interseção com

a grade de nível 100.0e0. Se quatro ou mais interseções resultam no nível 100.0e0, a geometria é promovida para o nível 1000.0e0. No nível 1000.0e0, as interseções devem ser fornecidas no índice espacial pois este é o nível mais alto possível.

A Figura 7 ilustra como quatro tipos diferentes de geometrias intersectam uma grade 10.0e. Todas as 23 interseções para as quatro geometrias são registradas no índice espacial.

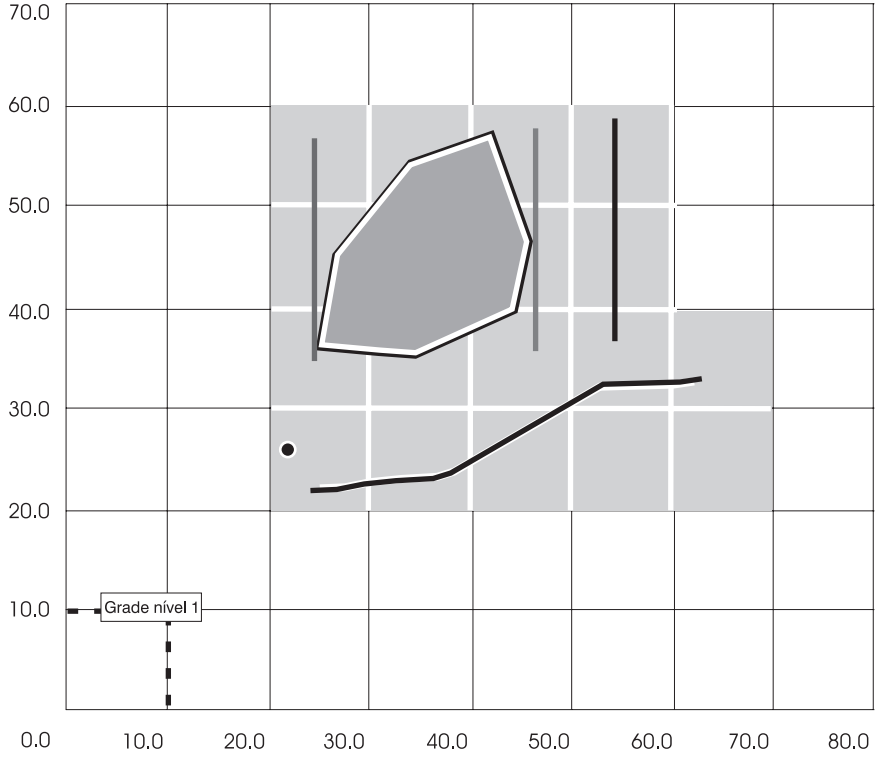


Figura 7. Aplicação de um nível de grade 10.0e0

A Tabela 38 relaciona as geometrias e suas interseções de grades correspondentes. Os envelopes de quatro tipos de geometrias diferentes intersectam a grade 10.0e. A coordenada mínima X e Y de cada célula de grade que ela intersecta é fornecida no índice espacial.

Tabela 38. As entradas da célula de grade 10.0e0 para os exemplos de geometrias

Geometria	Grade X	Grade Y
Polígono	20.0	30.0
Polígono	30.0	30.0

Tabela 38. As entradas da célula de grade 10.0e0 para os exemplos de geometrias (continuação)

Geometria	Grade X	Grade Y
Polígono	40.0	30.0
Polígono	20.0	40.0
Polígono	30.0	40.0
Polígono	40.0	40.0
Polígono	20.0	50.0
Polígono	30.0	50.0
Polígono	40.0	50.0
Linha vertical	50.0	30.0
Linha vertical	50.0	40.0
Linha vertical	50.0	50.0
Ponto	20.0	20.0
Linha Horizontal	20.0	20.0
Linha Horizontal	30.0	20.0
Linha Horizontal	40.0	20.0
Linha Horizontal	50.0	20.0
Linha Horizontal	60.0	20.0
Linha Horizontal	20.0	30.0
Linha Horizontal	30.0	30.0
Linha Horizontal	40.0	30.0
Linha Horizontal	50.0	30.0
Linha Horizontal	60.0	30.0

A Figura 8 na página 122 mostra como o número de interseções é notavelmente reduzido para oito através do acréscimo dos níveis de grade 30.0e0 e 60.0e0. Neste caso, o polígono identificado como geometria 1 é promovido para o nível de grade 30.0e0 e a linha identificada como geometria 4 é promovida para o nível de grade 60.0e0. Ao invés de nove e dez interseções que as geometrias tinham no nível 10.0e0, elas tem apenas duas após a promoção.

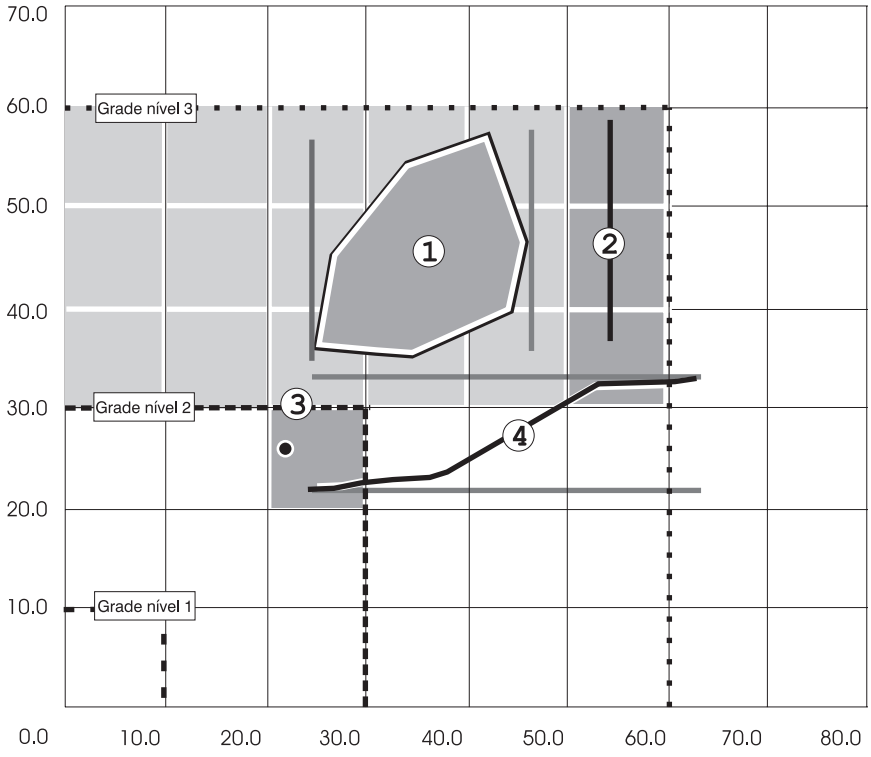


Figura 8. Efeito do acréscimo dos níveis de grade 30.0e0 e 60.0e0. O envelope do polígono identificado como a geometria 1 intersecta nove células da grade. O envelope da linha vertical identificado como geometria 2 intersecta três células da grade. O envelope do ponto identificado como geometria 3 intersecta apenas uma célula da grade. O envelope da linha identificada como geometria 4 intersecta dez células da grade.

O DB2 Spatial Extender obtém os parâmetros de nível de grade especificados na instrução CREATE INDEX e verifica cada objeto espacial para determinar as coordenadas e o número de blocos da grade em que o objeto existe. Na Figura 8, os níveis de grade 10.0e0, 30.0e0 e 60.0e0 são exibidos com pesos de linha sempre crescentes e diferentes tonalidades de cinza. As interseções da linha vertical e da célula do envelope de ponto são fornecidas no índice no nível de grade 10.0e0, pois ambas geram menos do que quatro interseções. O polígono intersecta nove células de grade 10.0e0 e é promovido para o nível de grade 30.0e0. Neste nível, o polígono intersecta duas células de grade, que são fornecidas no índice. A linha identificada como geometria 4 intersecta dez células de grade 10.0e0 e é promovida para o nível de grade 30.0e0. Ainda neste nível, ela intersecta seis células de grade, e é então novamente promovida para o nível de grade 60.0e0, onde ela gera duas interseções. Depois, as interseções da grade 60.0e0 da linha são fornecidas no índice. Se a linha tivesse gerado quatro ou mais interseções neste nível, elas ainda teriam sido fornecidas no índice pois este é o nível mais alto em que uma geometria pode ser promovida.

Tabela 39. As interseções das geometrias no índice de três fileiras

Geometria	Grade X	Grade Y
As interseções entre a linha vertical e o ponto no nível 1 (tamanho de grade 10.0e0)		
2	50.0	30.0
2	50.0	40.0
2	50.0	50.0
3	20.0	20.0
As interseções do polígono no nível 2 (tamanho de grade 30.0e0)		
1	0.0	30.0
1	30.0	30.0
As interseções da linha no nível 3 (tamanho de grade 60.0e0)		
4	0.0	0.0
4	60.0	0.0

Na verdade, o DB2 Spatial Extender não cria uma estrutura de grade do polígono de tipo algum. O DB2 Spatial Extender demonstra cada nível de grade, de forma paramétrica, definindo a origem no deslocamento de X,Y do sistema de referência espacial das colunas. Depois, ele estende a grade até o espaço da coordenada positiva. Utilizando uma grade paramétrica, o DB2 Spatial Extender gera, matematicamente, as interseções.

Diretrizes sobre o uso de um índice espacial

O DB2 Spatial Extender trabalha com índices espaciais para melhorar o desempenho das consultas espaciais. Considere a consulta espacial mais básica e provavelmente a mais popular, a consulta de caixa. Esta consulta solicita que o DB2 Spatial Extender retorne todas as geometrias que estão completamente ou parcialmente dentro de uma caixa definida pelo usuário. Quando não há um índice, o DB2 Spatial Extender precisa comparar todas as geometrias com a caixa. No entanto, com um índice, o DB2 Spatial Extender pode localizar todas as entradas do índice que possuem uma coordenada esquerda inferior maior ou igual a caixa e uma coordenada direita superior menor que ou igual a da caixa. Como o índice é ordenado por este sistemas de coordenadas, o DB2 Spatial Extender consegue obter rapidamente uma lista das geometrias candidatas. O processo que acabou de ser descrito é chamado de *primeiro passo*.

Um *segundo passo* determina se cada envelope do candidato intersecta a caixa. Uma geometria que se qualifica para o primeiro passo porque seu envelope de células de grade intersecta a caixa pode ela mesmo ter um envelope que não intersecta a caixa.

Um *terceiro passo* compara as coordenadas vigentes do candidato com a caixa para determinar se alguma parte da geometria está realmente dentro da caixa. Este último e um tanto complexo processo de comparação opera em uma lista de candidatos composta por um subconjunto da população total, que é significativamente reduzida pelos dois primeiros passos.

Todas as consultas espaciais realizam os três passos exceto pela função EnvelopesIntersect. Ela realiza apenas os dois primeiros passos. A função EnvelopesIntersect foi projetada para exibir operações que geralmente empregam suas próprias rotinas de recorte incorporadas e não requerem a granulosidade do terceiro passo.

Selecionando o tamanho da célula de grade

O shape irregular dos envelopes da geometria dificultam a seleção do tamanho da célula de grade. Devido a esta irregularidade, alguns envelopes de geometrias intersectam diversas grades, enquanto outros se ajustam dentro de uma única célula de grade. Inversamente, dependendo da distribuição espacial dos dados, algumas células de grade intersectam vários envelopes de geometrias.

Para que um índice espacial funcione bem, é essencial que o número e o tamanho corretos das grades sejam selecionados. Considere uma coluna espacial que contém geometrias dimensionadas uniformemente. Neste caso, um único nível de grade será suficiente. Comece com um tamanho de célula de grade que abranja o envelope médio da geometria. Ao testar sua aplicação, você pode perceber que aumentar o tamanho da célula de grade melhora o desempenho de suas consultas. Isto porque cada célula de grade contém mais geometrias e o primeiro passo tem condições de descartar mais rapidamente geometrias não-qualificadas. No entanto, você irá perceber que à medida que você continua a aumentar o tamanho da célula, o desempenho começa a decair. Isto porque eventualmente o segundo passo terá que sustentar mais candidatos.

Selecionando o número de níveis

Se os objetos a serem indexados são aproximadamente do mesmo tamanho relativo, você pode usar um único nível de grade. Embora isto seja verdade, nem todas as colunas irão conter geometrias do mesmo tamanho relativo. Geralmente, as geometrias de colunas espaciais podem ser agrupadas em diversos intervalos de tamanho. Por exemplo, considere uma rede rodoviária na qual as geometrias são divididas em ruas, avenidas e estradas. As ruas são quase todas do mesmo comprimento e podem ser agrupadas em um intervalo de tamanho. Isto também é verdadeiro para as avenidas e estradas. Sendo assim, as ruas, representando o um intervalo de tamanho, poderiam ser agrupadas no primeiro nível de grade, as redes rodoviárias no segundo e as estradas no terceiro nível de grade. Outro exemplo inclui uma coluna de lote de município que contém grupos de pequenos lotes urbanos cercados por

lotes rurais maiores. Neste caso, há dois intervalos de tamanho e dois níveis de grade, uma para os pequenos lotes urbanos e outro para os lotes rurais maiores. Estas situações são muito comuns e exigem o uso de uma grade de multiníveis.

Para selecionar o tamanho de célula para cada nível de grade, selecione os tamanhos da célula de grade que são um pouco maiores do que cada intervalo de tamanho. Teste o índice realizando consultas junto à coluna espacial.

Cada nível adicional requer uma pesquisa de índice extra. Tente ajustar os tamanhos de grade um pouco para cima ou para baixo para determinar se uma melhoria considerável no desempenho pode ser obtida.

Capítulo 13. Geometrias e funções espaciais associadas

Este capítulo discute as unidades de informações, denominadas *geometrias*, que consistem em coordenadas e simbolizam recursos geográficos. Apresenta também as funções espaciais que tomam geometrias como entrada e retornam resultados para ajudá-lo a analisar recursos geográficos e deslocar dados espaciais entre sistemas informativos geográficos. Os tópicos abrangidos são:

- A natureza das geometrias
- As propriedades das geometria; funções que retornam informações relacionadas a estas propriedades
- Geometrias instanciáveis; funções que operam nelas
- Funções que:
  - Mostram relações e comparações entre recursos geográficos
  - Geram geometrias
  - Convertem valores de geometria em formatos para importação e exportação

Sobre geometrias

O dicionário Oxford (Oxford American Dictionary) define *geometria* como "ramo da matemática que estuda as propriedades e relações das linhas, ângulos, superfícies e sólidos." Em 11 de agosto de 1997, o Open GIS Consortium Inc. (OGC) em sua publicação, *Open GIS Features for ODBC (SQL) Implementation Specification*, criou outra definição para o termo. A palavra *geometria* foi selecionada para designar os recursos geométricos que, desde o último milênio, ou mais, os cartógrafos usaram para mapear o mundo. Uma definição muito abstrata deste novo significado de geometria pode ser "um ponto ou conjunto de pontos que simbolizam um recurso na terra."

No DB2 Spatial Extender, uma definição *operacional* de geometria poderia ser "um modelo de recursos geográfico." O modelo pode ser expresso em termos das coordenadas do recurso e também, em alguns casos, de um símbolo visual. O modelo transmite informações; por exemplo, as coordenadas identificam a posição do recurso com relação aos pontos fixos da referência e o símbolo descreve seu formato. Além disso, o modelo pode ser usado para produzir informações; por exemplo, a função ST_Overlaps pode tomar as coordenadas de duas regiões próximas como entrada e retornar informações indicando se as regiões estão sobrepostas.

As coordenadas de um recurso que uma geometria simboliza são consideradas *propriedades* da geometria. Vários tipos de geometrias também têm outras propriedades, por exemplo:

- Um *interior* representa o conteúdo do recurso que a geometria simboliza.
- Um *exterior* representa o espaço ao redor do recurso.
- Um *limite* representa a demarcação em que o conteúdo termina e o espaço adjacente começa.

Estas propriedades adicionais serão discutidas em “Propriedades de geometrias e funções associadas” na página 129.

As geometrias suportadas pelo DB2 Spatial Extender formam uma hierarquia, mostrada na Figura 9. Seis membros da hierarquia são instanciáveis, eles podem ser expressos como símbolos visuais, que são mostrados na figura.

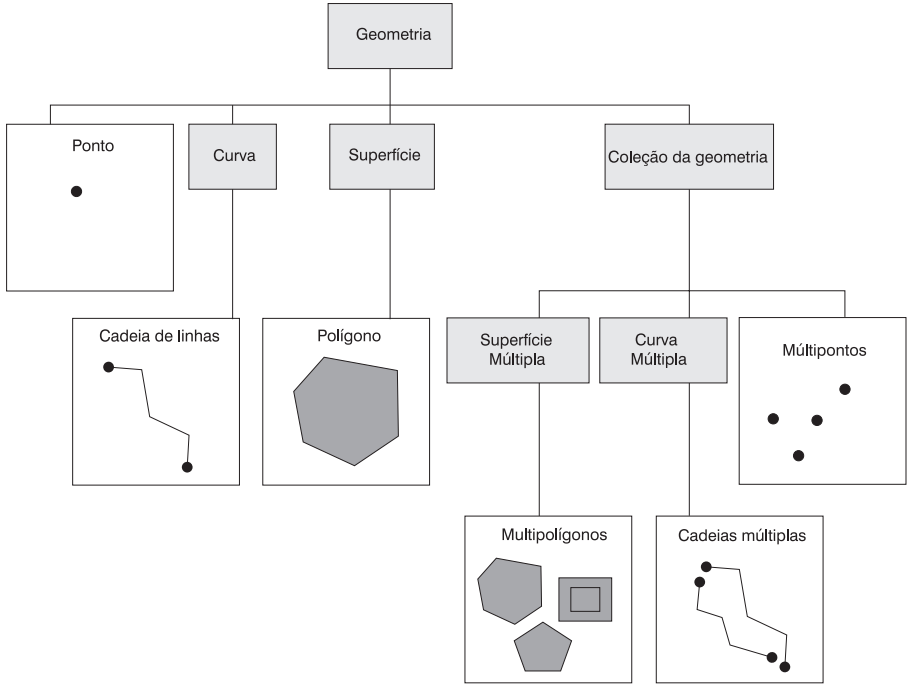


Figura 9. Hierarquia das geometrias suportadas pelo DB2 Spatial Extender. As geometrias instanciáveis podem ser expressas como símbolos visuais. Tais símbolos são mostrados abaixo dos nomes destas geometrias.

Como Figura 9 indica, uma superclasse chamada *geometria* é a raiz da hierarquia. As classes filhas estão divididas em duas categorias: as classes filhas da geometria base e as classes filhas da coleção homogênea. As geometrias base incluem:

- *Pontos*, que simbolizam recursos discretos que são captados como ocupantes do local em que uma linha da coordenada leste / oeste (como um paralelo) intercepta uma linha da coordenada norte / sul (como um meridiano). Por exemplo, supondo-se que a notação de um mapa em escala grande mostra que cada cidade no mapa está localizada na interseção de um paralelo e de um meridiano. Nesta escala, cada cidade seria representada por um ponto.
- *Cadeias de linhas*, que simbolizam recursos geográficos lineares (por exemplo, ruas, canais e pipelines).
- *Polígonos*, os quais simbolizam recursos geográficos multifacetados (por exemplo, distritos de preservação, florestas e habitats naturais).

As coleções homogêneas incluem:

- *Multipontos*, que simbolizam recursos de várias partes cujos componentes estão localizados cada um na interseção de uma linha da coordenada leste / oeste e uma linha da coordenada norte / sul (por exemplo, uma cadeia de ilhas cujos membros estejam situados na interseção de um paralelo e um meridiano).
- *Cadeias de linhas múltiplas*, que simbolizam recursos de várias partes compostos de unidades ou componentes lineares (por exemplo, sistemas fluviais e sistemas rodoviários).
- *Multipolígonos*, que simbolizam recursos de várias partes compostos de unidades ou componentes multifacetados (por exemplo, as fazendas coletivas de uma região específicas ou um sistema de lagos).

Como seu nome implica, as coleções homogêneas são coleções de geometrias básicas. Além de compartilhar as propriedades da geometria básica, as coleções homogêneas possuem algumas propriedades próprias também.

Os tipos de dados espaciais suportados pelo DB2 Spatial Extender são implementações das geometrias mostradas na Figura 9 na página 128. Para obter uma descrição destes tipos de dados, consulte “Sobre tipos de dados espaciais” na página 33.

Propriedades de geometrias e funções associadas

Esta seção descreve as propriedades das geometrias e as funções espaciais que estão associadas à estas propriedades. A seção começa com as propriedades do núcleo:

- A qual classe uma geometria pertence
- Coordenadas X e Y

Esta seção também explica:

- Coordenadas Z
- Medidas
- Um interior, limite e exterior da geometria

- A qualidade de ser simples ou não-simples
- A qualidade de ser vazio ou não-vazio
- Um envelope de geometria
- Dimensão
- O identificador de uma geometria associada ao sistema de referência espacial

Classe

Cada geometria pertence a uma classe na hierarquia mostrada na Figura 9 na página 128. Conforme indicado em "Sobre geometrias" na página 127, seis classes filhas na hierarquia — pontos, cadeias de linhas, polígonos, multipontos, cadeias de linhas múltiplas e multipolígonos — são instanciáveis. A superclasse e outras classes filhas não são instanciáveis.

A função ST_GeometryType toma uma geometria e retorna a subclasse instanciável no formato de uma cadeia de caracteres. Para obter mais informações, consulte "ST_GeometryType" na página 222.

A função ST_IsValid toma uma geometria que foi atribuída a um tipo de dados ST_Geometry. A função retorna 1 (VERDADEIRO) se a geometria estiver válida e 0 (FALSO) se não estiver válida. Para obter mais informações, consulte "ST_IsValid" na página 239.

Coordenadas X e Y

Um *valor da coordenada X* indica um local que é relativo a um ponto de referência a leste ou oeste. Um *valor da coordenada Y* indica um local que é relativo a um ponto de referência ao norte ou sul. Para mais informações, consulte "A natureza dos dados espaciais" na página 6 e "Sobre sistemas de coordenadas e de referência espacial" na página 25.

Coordenadas Z

Algumas geometrias apresentam uma altitude ou profundidade associada. Cada um dos pontos que formam a geometria de um recurso podem incluir uma coordenada Z opcional que representa uma altitude ou profundidade normal à superfície da terra.

A função do predicado Is3d toma uma geometria e retorna 1 (VERDADEIRO) se a função tiver coordenadas Z e, do contrário, 0 (FALSO). Para obter mais informações, consulte "Is3d" na página 169.

Medidas

Uma medida é um valor que expressa informações sobre um recurso geográfico que é armazenado junto às coordenadas que definem a localização do recurso. Por exemplo, suponha que você está representando sistemas de transporte no GIS. Se desejar que sua aplicação processe seus valores de processo que indicam distâncias lineares ou postes, você pode armazenar estes

valores junto às coordenadas que definem as localizações dos sistemas. As medidas são armazenadas como números de precisão dupla.

O predicado IsMeasured toma uma geometria e retorna 1 (VERDADEIRO) se contiver medidas e, do contrário, 0 (FALSO). Para obter mais informações, consulte "IsMeasured" na página 170.

Interior, limite e exterior

Todas as geometrias ocupam uma posição no espaço definido por seu interior, limite e exterior. O exterior de uma geometria é todo o espaço não ocupado pela geometria. O limite de uma geometria serve como a interface entre seu interior e exterior. O interior é o espaço ocupado pela geometria. As classes filhas herdam as propriedades do interior e do exterior diretamente, embora a propriedade de limite seja diferente para cada.

A função ST_Boundary toma uma geometria e retorna uma que represente o limite da geometria de origem. Para obter mais informações, consulte "ST_Boundary" na página 192.

Simples ou não-simples

Algumas classes filhas de geometria (cadeias de linhas, multipontos e cadeias de linhas múltiplas são simples ou não-simples. Uma subclasse é simples se obedecer a todas as regras impostas à subclasse e a não-simples é o oposto. Uma cadeia de linhas é simples se não fizer interseção com seu interior. Um multiponto é simples se nenhum de seus elementos ocupar o mesmo espaço da coordenada. Uma cadeia de linhas múltiplas é simples se nenhum interior de seu elemento fizer interseção pelo seu próprio interior.

A função do predicado ST_IsSimple toma uma geometria e retorna 1 (VERDADEIRO) se a geometria for simples e, caso contrário, 0 (FALSO). Para obter mais informações, consulte "ST_IsSimple" na página 238.

Vazio ou não-vazio

Uma geometria é vazia se não possuir pontos. O envelope, o limite, o interior e o exterior de uma geometria vazia são NULOS. Uma geometria vazia é sempre simples e pode ter coordenadas Z ou medidas. Cadeias de linhas e cadeias de linhas múltiplas vazias têm comprimento 0. Os polígonos vazios e os multipolígonos têm área 0.

A função do predicado ST_IsEmpty toma uma geometria e retorna 1 (VERDADEIRO) se a geometria estiver vazia e, do contrário, 0 (FALSO). Para obter mais informações, consulte "ST_IsEmpty" na página 234.

Envelope

O envelope de uma geometria é a geometria limitada formada por coordenadas mínimas e máximas (X,Y). Com as seguintes exceções, os envelopes da maioria das geometrias formam um retângulo de limite:

- O envelope de um ponto é o próprio ponto, pois suas coordenadas mínimas e máximas são as mesmas.
- O envelope de uma cadeia de linha horizontal ou vertical é uma cadeia de linha representada pelo limite (os nós de extremidade) da cadeia de linha fonte.

A função ST_Envelope toma uma geometria e retorna uma geometria limitada, que representa seu envelope. Para obter mais informações, consulte "ST_Envelope" na página 212.

Dimensão

Uma geometria pode ter uma dimensão de 0, 1 ou 2. As dimensões estão relacionadas da seguinte forma:

0 Não tem comprimento nem área

1 Tem comprimento

2 Contém área

As classes filhas ponto e multiponto possuem dimensão zero. Os pontos representam recursos dimensionais que podem ser modelados com uma única coordenada, ao passo que as classes filhas de multiponto representam dados que devem ser modelados com um grupo de coordenadas desconectadas.

As classes filhas cadeia de linhas e cadeia de linhas múltiplas têm dimensão um. Elas armazenam segmentos de rodovia, extensões de rios e quaisquer outros recursos que sejam lineares na natureza.

As classes filhas polígono e multipolígono possuem dimensão dois. Os recursos cujos perímetros englobem uma área definível, como florestas, pedaços de terra e rios podem ser cedidos pelo tipo de dados polígono ou multipolígono.

A dimensão é importante não somente como uma propriedade da subclasse, como também desempenha uma parte na determinação da relação espacial de dois recursos. A dimensão dos recursos resultantes determina se operação foi bem-sucedida. O DB2 Spatial Extender examina a dimensão dos recursos para determinar como serão comparados.

A função ST_Dimension toma uma geometria e retorna sua dimensão como um inteiro. Para obter mais informações, consulte "ST_Dimension" na página 206.

Identificador do sistema de referência espacial

O sistema de referência espacial identifica a transformação de coordenadas para cada geometria.

Todos os sistemas de referência espaciais reconhecidos pelo banco de dados podem ser acessados através da view do catálogo DB2GSE.SPATIAL_REF_SYS. Para obter informações sobre esta view, consulte “Capítulo 11. Views do catálogo” na página 113.

A função ST_SRID toma uma geometria e retorna seu identificador de referência espacial como um inteiro. Para obter mais informações, consulte “ST_SRID” na página 272.

A função ST_Transform atribui uma geometria a um sistema de referência espacial diferente do sistema de referência espacial no qual a geometria está atribuída atualmente. Para obter mais informações, consulte “ST_Transform” na página 277.

Geometrias instanciáveis e funções associadas

Esta seção descreve as seis classes filhas das geometrias instanciáveis e as funções que operam nelas. As classes filhas são:

- Pontos
- Cadeias de linhas
- Polígonos
- Multipontos
- Cadeias de linhas múltiplas
- Multipolígonos

Para obter ilustrações da hierarquia a que pertencem estas classes filhas e dos símbolos visuais associados a elas, consulte Figura 9 na página 128.

Pontos

Um ponto indica uma geometria dimensional zero que ocupa uma única localização no espaço da coordenada. Um ponto inclui uma coordenada X e uma coordenada Y que definem esta localização. Também pode incluir uma coordenada Z e uma medida.

Um ponto é simples e possui um limite NULL. Os pontos geralmente são utilizados para definir recursos como poços de petróleo, pontos de referência e elevações.

Funções que operam exclusivamente na subclasse do ponto:

ST_Point
: Toma uma coordenada X, sua coordenada Y associada e o identificador do sistema de referência espacial, ao qual pertencem estas coordenadas, e retorna o ponto que as coordenadas definem. Para obter mais informações, consulte “ST_Point” na página 263.

ST_CoordDim
Retorna um valor que indica quais coordenadas um ponto contém e se contém uma medida também. Este valor é chamado de *dimensão da coordenada*. As dimensões das coordenadas possíveis são:

2 O ponto consiste numa coordenada X e uma coordenada Y.

3 O ponto consiste numa coordenada X, uma coordenada Y e uma coordenada Z.

4 O ponto consiste numa coordenada X, uma coordenada Y, uma coordenada Z e uma medida.

Para obter mais informações, consulte “ST_CoordDim” na página 201.

ST_PointFromText
Toma uma representação de texto binário reconhecida (well-known text - WKT) de OGC de um ponto e retorna o ponto. Para obter mais informações, consulte “ST_PointFromText” na página 260.

ST_X Retorna um valor da coordenada X do tipo de dados ST_Point como um número de precisão dupla. Para obter mais informações, consulte “ST_X” na página 283.

ST_Y Retorna um valor da coordenada Y do tipo de dados ST_Point como um número de precisão dupla. Para obter mais informações, consulte “ST_Y” na página 284.

Z Retorna um valor da coordenada Z do tipo de dados ST_Point como um número de precisão dupla. Para obter mais informações, consulte “Z” na página 285.

M Retorna uma medida do tipo de dados ST_Point como um número de precisão dupla. Para obter mais informações, consulte “M” na página 177.

Cadeias de linhas

Uma cadeia de linha é um objeto unidimensional armazenando como uma seqüência de pontos que definem um caminho linear interpolado. A cadeia de linhas é simples se não fizer interseção com seu interior. Os nós de extremidade (o limite) de uma cadeia de linhas fechada ocupam o mesmo ponto no espaço. Uma cadeia de linha indica um anel se estiver fechada e seu interior não se interceptar. Além das outras propriedades herdadas da geometria da superclasse, as cadeias de linhas têm comprimento. Elas são geralmente usadas para definir recursos lineares como estradas, rios e linhas de energia.

Uma cadeia de linha simples cujo ponto inicial e final sejam os mesmos é denominada *anel*.

Os nós de extremidade geralmente formam um limite de uma cadeia de linha, a menos que as linhas estejam fechadas, tornando o limite NULO. O interior de uma cadeia de linha é o caminho conectado que fica entre os nós de extremidade, a menos que esteja fechada, tornando o interior contínuo.

Funções que operam em cadeias de linhas:

ST_StartPoint
Toma uma cadeia de linha e retorna seu primeiro ponto. Para obter mais informações, consulte “ST_StartPoint” na página 273.

ST_EndPoint
Toma uma cadeia de linha e retorna seu último ponto. Para obter mais informações, consulte “ST_Endpoint” na página 211.

ST_PointN
Toma uma cadeia de linha e um índice ao ponto *n* e retorna esse ponto. Para obter mais informações, consulte “ST_PointN” na página 264.

ST_Length
Toma uma cadeia de linha e retorna seu comprimento como número de precisão dupla. Para obter mais informações, consulte “ST_Length” na página 241.

ST_NumPoints
Toma uma cadeia de linha e retorna o número de pontos na sua seqüência como um inteiro. Para obter mais informações, consulte “ST_NumPoints” na página 255.

ST_IsRing
Toma uma cadeia de linha e retorna 1 (VERDADEIRO) se a cadeia de linha for um anel e, do contrário, 0 (FALSO). Para obter mais informações, consulte “ST_IsRing” na página 236.

ST_IsClosed
Toma uma cadeia de linha e retorna 1 (VERDADEIRO) se a cadeia de linha estiver fechada e, do contrário, 0 (FALSO). Para obter mais informações, consulte “ST_IsClosed” na página 232.

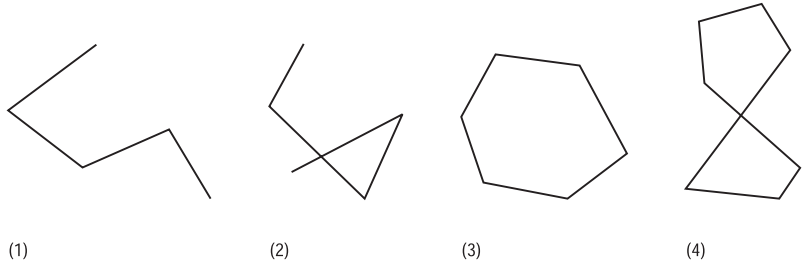


Figura 10. Objetos da cadeia de linha.

1. Uma cadeia de linha não-fechada simples.
2. Uma cadeia de linha não-simples, não-fechada.
3. Uma cadeia de linha simples fechada e, portanto, um anel.
4. Uma cadeia de linha fechada, não-simples. Não é um anel.

Polígonos

Um polígono é uma superfície bidimensional armazenada como uma seqüência de pontos definindo seu anel de limite exterior e 0 ou mais anéis interiores. Os anéis de um polígono não podem se sobrepor. Portanto, por definição, os polígonos sempre serão simples. Geralmente eles definem pedaços de terra, rios e outros recursos que possuam uma extensão espacial.

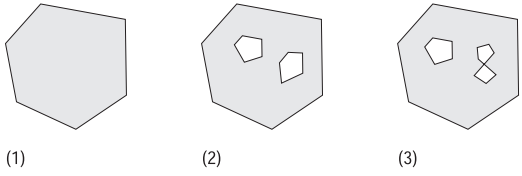


Figura 11. Polígonos.

1. Um polígono cujo limite esteja definido por um anel exterior.
2. Um polígono cujo limite esteja definido por um anel exterior e dois anéis interiores. A área dentro dos anéis interiores fazem parte do exterior dos polígonos.
3. Um polígono permitido, pois os anéis fazem interseção num único ponto da tangente.

Os anéis externo e interno definem o limite de um polígono e o espaço entre os anéis define o interior do polígono. Os anéis de um polígono podem fazer interseção num ponto da tangente mas nunca se cruzarem. Além das outras propriedades herdadas da geometria da superclasse, os polígonos possuem área.

Funções que operam em polígonos:

ST_Area
Toma um polígono e retorna sua área como um número de precisão dupla. Para obter mais informações, consulte “ST_Area” na página 188.

ST_ExteriorRing
Toma um polígono e retorna seu anel externo como uma cadeia de linha. Para obter mais informações, consulte “ST_ExteriorRing” na página 215.

ST_NumInteriorRing
Toma um polígono e retorna o número de anéis internos que ele contém. Para obter mais informações, consulte “ST_NumInteriorRing” na página 254.

ST_InteriorRingN
Toma um polígono e um índice e retorna o anel interno *n* como uma cadeia de linha. Para obter mais informações, consulte “ST_InteriorRingN” na página 224.

ST_Centroid
Toma um polígono e retorna um ponto que é o centro da extensão do polígono. Para obter mais informações, consulte “ST_Centroid” na página 196.

ST_PointOnSurface
Toma um polígono e retorna um ponto que tem garantia de estar na superfície di polígono. Para obter mais informações, consulte “ST_PointOnSurface” na página 265.

ST_Perimeter
Toma um polígono e retorna o perímetro de sua superfície. Para obter mais informações, consulte “ST_Perimeter” na página 259.

Multipontos

Um multiponto é uma coleção de pontos e, como seus elementos, possui dimensão 0. Um multiponto será simples se nenhum elemento seu ocupar o mesmo espaço de coordenadas. O limite de um multiponto é NULO. Os multipontos podem ser usados para definir fenômenos como padrões de difusão aérea e incidentes de um surto epidêmico.

Funções que operam em multipontos:

ST_NumGeometries
Toma uma coleção homogênea e retorna o número de elementos da geometria básica que contém. Para obter mais informações, consulte “ST_NumGeometries” na página 253.

ST_GeometryN

Toma uma coleção homogênea e um índice e retorna a geometria básica nth. Para obter mais informações, consulte “ST_GeometryN” na página 221.

Cadeias de linhas múltiplas

Uma cadeia de linhas múltiplas é uma coleção de cadeias de linhas. As cadeias de linhas múltiplas são simples se elas fizerem interseção nos pontos de extremidade dos elementos da cadeia de linha. As cadeias de linhas múltiplas serão não-simples se os interiores dos elementos da cadeia de linhas fizerem interseção.

O limite de uma cadeia de linhas múltiplas será o ponto de extremidade não-interceptado dos elementos da cadeia de linhas. A cadeia de linhas múltiplas será fechada se todos os elementos da cadeia de linhas estiverem fechados. O limite de uma cadeia de linhas múltiplas será NULO se todos os pontos de extremidade de todos os elementos fizerem interseção. Além das outras propriedades herdadas da geometria da superclasse, as cadeias de linhas múltiplas têm comprimento. As cadeias de linhas múltiplas são usadas para definir fluxos ou redes de estradas.

Funções que operam em cadeias de linhas múltiplas:

ST_Length

Toma uma cadeia de linhas múltiplas e retorna o comprimento cumulativo de todos os elementos da cadeia de linhas como um número de precisão dupla. Para obter mais informações, consulte “ST_Length” na página 241.

ST_IsClosed

Toma uma cadeia de linhas múltiplas e retorna 1 (VERDADEIRO) se a cadeia de linhas múltiplas estiver fechada e, do contrário, 0 (FALSO). Para obter mais informações, consulte “ST_IsClosed” na página 232.

ST_NumGeometries

Toma uma coleção homogênea e retorna o número de elementos da geometria básica que contém. Para obter mais informações, consulte “ST_NumGeometries” na página 253.

ST_GeometryN

Toma uma coleção homogênea e um índice e retorna a geometria básica nth. Para obter mais informações, consulte “ST_GeometryN” na página 221.

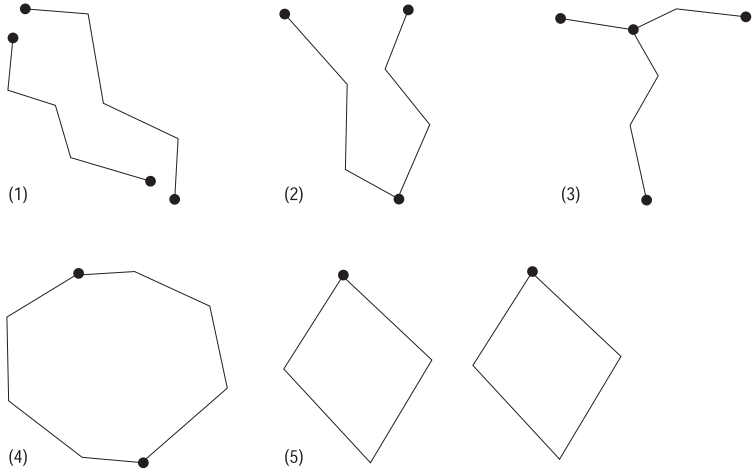
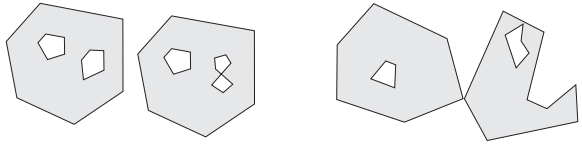


Figura 12. Cadeias de linhas múltiplas.

1. Uma cadeia de linhas múltiplas cujo limite esteja definido pelos quatro pontos de extremidade de seus dois elementos da cadeia de linhas.
2. Uma cadeia de linhas múltiplas simples somente porque os pontos de extremidade dos elementos da cadeia de linhas fizerem interseção. O limite é definido por dois pontos de extremidade que não fazem interseção.
3. Uma cadeia de linha não-simples porque o interior de um dos elementos da cadeia de linha está interceptado. O limite desta cadeia de linhas múltiplas está definido pelos quatro pontos de extremidade, inclusive o ponto de interseção.
4. Uma cadeia de linhas múltiplas não-fechada. Não está fechada porque suas cadeias de linha de elementos não estão fechadas. É simples porque nenhum interior das cadeias de linhas de elementos fazem interseção.
5. Uma cadeia de linhas múltiplas fechada. Está fechada porque todos os elementos estão fechados. É simples porque nenhum de seus elementos fazem interseção nos interiores.

Multipolígonos

O limite de um multipolígono é o comprimento cumulativo dos anéis externo e interno de seu elemento. O interior de um multipolígono é definido como o interior cumulativo dos polígonos do elemento. O limite dos elementos de um multipolígono podem fazer interseção somente no ponto de tangência. Além das outras propriedades herdadas da geometria da superclasse, os multipolígonos possuem área. Os multipolígonos definem recursos como uma camada da florestas ou uma parte não-contígua de terra, como uma cadeia de ilhas.



(1) (2)

Figura 13. Multipolígonos.

1. Um multipolígono com dois elementos do polígono. O limite é definido por dois anéis externos e três internos.
2. Um multipolígono com dois elementos do polígono. O limite é definido por dois anéis externos e dois internos. Os dois elementos do polígono fazem interseção num ponto da tangente.

Funções que operam em multipolígonos:

ST_Area
Toma um multipolígono e retorna a área cumulativa dos elementos do polígono como um número de precisão dupla. Para obter mais informações, consulte "ST_Area" na página 188.

ST_Centroid
Toma um multipolígono e retorna um ponto que é seu centro geométrico dimensionado. Para obter mais informações, consulte "ST_Centroid" na página 196.

ST_NumGeometries
Toma uma coleção homogênea e retorna o número de elementos da geometria básica que contém. Para obter mais informações, consulte "ST_NumGeometries" na página 253.

ST_GeometryN
Toma uma coleção homogênea e um índice e retorna a geometria básica nth. Para obter mais informações, consulte "ST_GeometryN" na página 221.

Funções que mostram relações e comparações, geram geometrias, e convertem formatos de valores

As seções precedentes apresentam três categorias de funções espaciais:

- Funções associadas às propriedades da geometria
- Funções associadas a geometrias específicas

Esta seção apresenta mais três categorias:

- Funções que determinam formas nas quais os recursos geográficos se relacionam ou se comparam
- Funções que geram novas geometrias

- Funções que convertem os valores de uma geometria em um formato que pode ser importado ou exportado

Funções que mostram relações ou comparações entre recursos geográficos

Várias funções espaciais retornam informações sobre formas em que os recursos geográficos se relacionam ou se comparam. A maioria destas funções, denominadas *predicados*, são funções booleanas. Esta seção descreve os predicados em geral e, em seguida, discute cada função individualmente.

Funções do predicado

As funções de predicado retornam 1 (VERDADEIRO) se uma comparação atender aos critérios da função, ou 0 (FALSO) se a comparação falhar. Os predicados que testam uma relação espacial comparam pares de geometrias que podem ser de um tipo ou dimensão diferentes.

Os predicados comparam as coordenadas X e Y das geometrias submetidas. As coordenadas Z e a medida (se existirem) serão ignoradas. Isto permite que as geometrias tenham coordenadas Z ou medida para comparação com aquelas que não têm.

O *Modelo de Interseção 9 Dimensionalmente Estendido (DE-9IM*) [1]é uma abordagem matemática que define a relação espacial pair-wise entre geometrias de tipos e dimensões diferentes. Este modelo expressa relações espaciais entre todos os tipos de geometria como interseções pair-wise de seu interior, limite e exterior e consideração com a dimensão das interseções resultantes.

Dadas as geometrias *a* e *b*: I(*a*), B(*a*) e E(*a*) representam o interior, o limite e o exterior de *a*. E, I(b), B(b) e E(b) representam o interior, o limite e o exterior de *b*. As interseções de I(a), B(a) e E(a) com I(b), B(b) e E(b) produzem uma matriz 3 por 3. Cada interseção pode resultar em geometrias de dimensões diferentes. Por exemplo, a interseção dos limites de dois polígonos consiste num ponto e numa cadeia de linha. Nesse caso, a função dim retornará a dimensão máxima de 1.

A função dim retorna um valor de 1, 0, 1 ou 2. 1 corresponde ao conjunto nulo ou dim(null), que retorna quando nenhuma interseção é encontrada.

---

1. O DE-91M foi desenvolvido por Clementini e Felice, que estenderam dimensionalmente o Modelo de Interseção 9 de Egenhofer e Herring. O DE-91M é colaboração de quatro autores, Clementini, Eliseo, Di Felice e van Osstrom. Eles publicaram o modelo em ″A Small Set of Formal Topological Relationships Suitable for End-User Interaction,″ D. Abel e B.C. Ooi (Ed.), *Advances in Spatial Database — no Terceiro Simpósio Internacional. SSD '93.* LNCS 692. Páginas 277 a 295. O modelo de Interseção 9 desenvolvido por Springer-Verlag Singapore (1993) Egenhofer M.J. e Herring, J., foi publicado em ″Categorizing binary topological relationships between regions, lines, and points in geographic databases,″ *Tech. Report, Department of Surveying Engineering,* Universidade de Maine, Orono, ME 1991.

	Interior	Limite	Exterior
Interior	dim(I(a) ∩ I(b))	dim(I(a) ∩ B(b))	dim(I(a) ∩ E(b))
Limite	dim(B(a) ∩ I(b))	dim(B(a) ∩ B(b))	dim(B(a) ∩ E(b))
Exterior	dim(E(a) ∩ I(b))	dim(E(a) ∩ B(b))	dim(E(a) ∩ E(b))

Os resultados dos predicados da relação espacial podem ser entendidos ou verificados através da comparação de resultados do predicado com a matriz padrão que representa os valores aceitáveis para DE-9IM.

A matriz padrão contém os valores aceitáveis para cada célula da matriz de interseção. Os valores padrão possíveis são:

T Deve existir uma interseção, dim = 0, 1 ou 2.

F Não deve existir uma interseção, dim = -1.

***** Não importa se existe uma interseção, dim = -1, 0, 1 ou 2.

0 Deve existir uma interseção e sua dimensão máxima deve ser 0, dim = 0.

1 Deve existir uma interseção e sua dimensão máxima deve ser 1, dim = 1.

2 Deve existir uma interseção e sua dimensão máxima deve ser 2, dim = 2.

Por exemplo a seguinte matriz padrão do predicado ST_Within inclui os valores T, F e *.

Tabela 40. Matriz para ST_Within. A matriz padrão do predicado ST_Within para combinações de geometria.

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	T	*	F
	Limite	*	*	F
	Exterior	*	*	*

O predicado ST_Within retorna VERDADEIRO quando os interiores das duas geometrias fazem interseção e quando o interior e o limite de *a* não faz interseção com o exterior de *b*. Todas as outras condições não importam.

Cada predicado tem pelo menos uma matriz padrão, mas alguns exigem mais que uma para descrever as relações de várias combinações de tipo de geometria.

ST_Equals

ST_Equals retorna 1 (VERDADEIRO) se duas geometrias do mesmo tipo tiverem valores idênticos das coordenadas X,Y.

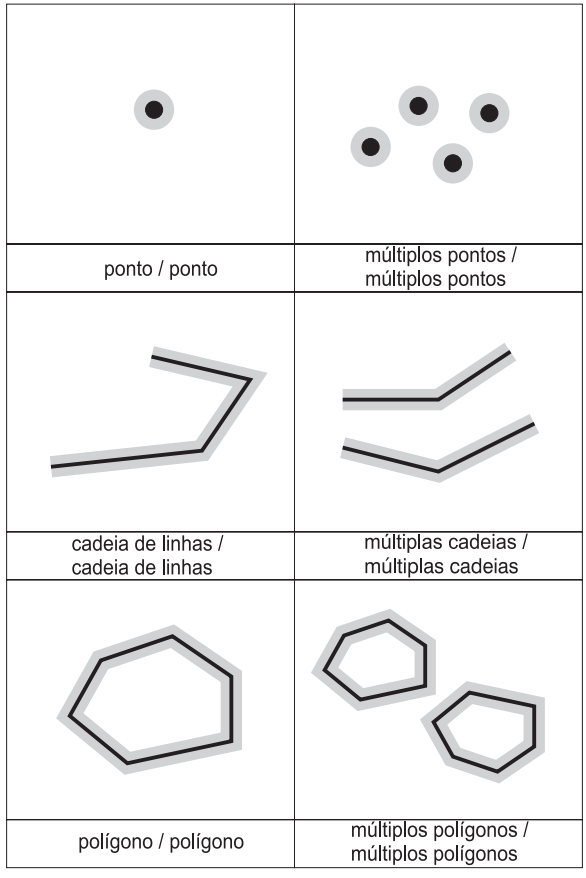


Figura 14. ST_Equals. As geometrias serão iguais se possuírem coordenadas X,Y correspondentes.

Tabela 41. Matriz para igualdade. A matriz padrão DE-9IM para igualdade assegura que os interiores fazem interseção e que nenhuma parte ou limite interior da geometria faz interseção com o exterior de outra.

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	T	*	F
	Limite	*	*	F
	Exterior	F	F	*

Para obter mais informações, consulte “ST_Equals” na página 214.

ST_OrderingEquals

ST_OrderingEquals compara duas geometrias e retorna 1 (VERDADEIRO) se as geometrias forem iguais e as coordenadas estiverem na mesma ordem; do contrário, retornará 0 (FALSO). Para obter mais informações, consulte “ST_OrderingEquals” na página 256.

ST_Disjoint

ST_Disjoint retorna 1 (VERDADEIRO) se a interseção das duas geometrias for um conjunto vazio.

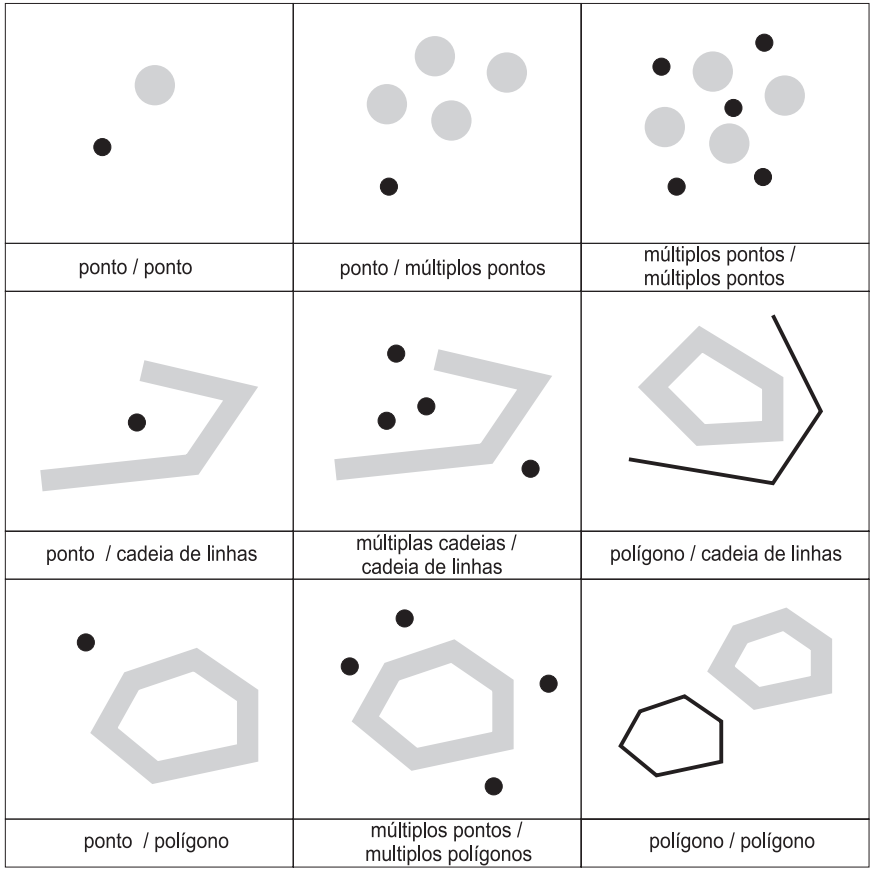


Figura 15. ST_Disjoint. As geometrias serão desunidas se não fizerem interseção entre si de alguma forma.

Tabela 42. Matriz para ST_Disjoint. A matriz padrão do predicado ST_Disjoint simples indica que nem o interior nem o limite da geometria fazem interseção.

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	F	F	*
	Limite	F	F	*
	Exterior	*	*	*

Para obter mais informações, consulte “ST_Disjoint” na página 208.

ST_Intersects

ST_Intersects retorna 1 (VERDADEIRO) se a interseção não resultar em um conjunto vazio. A interseção retorna o resultado oposto exato de ST_Disjoint.

O predicado ST_Intersects retorna VERDADEIRO se as condições de quaisquer matrizes padrão seguintes retornarem VERDADEIRO.

Tabela 43. Matriz para ST_Intersects (1). O predicado ST_Intersects retorna VERDADEIRO se os interiores das duas geometrias fizerem interseção.

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	T	*	*
	Limite	*	*	*
	Exterior	*	*	*

Tabela 44. Matriz para ST_Intersects (2). O predicado ST_Intersects retorna VERDADEIRO se o limite da primeira geometria fizer interseção com o limite da segunda.

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	*	T	*
	Limite	*	*	*
	Exterior	*	*	*

Tabela 45. Matriz para ST_Intersects (3). O predicado ST_Intersects retorna VERDADEIRO se o limite da primeira geometria fizer interseção com o interior da segunda.

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	*	*	*
	Limite	T	*	*
	Exterior	*	*	*

Tabela 46. Matriz para ST_Intersects (4). O predicado ST_Intersects retorna VERDADEIRO se os limites de qualquer geometria fizerem interseção.

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	*	*	*
	Limite	*	T	*
	Exterior	*	*	*

Para obter mais informações, consulte “ST_Intersects” na página 231.

EnvelopesIntersect

Esta função retorna 1 (VERDADEIRO) se os envelopes das duas geometrias fizerem interseção. É uma função de conveniência que implementa eficientemente ST_Intersects (`ST_Envelope(g1),ST_Envelope(g2)`). Para obter mais informações, consulte “EnvelopesIntersect” na página 167.

ST_Touches

ST_Touches retorna 1 (VERDADEIRO) se nenhum dos pontos comuns a ambas as geometrias fizerem interseção com os interiores das duas. Pelo menos uma geometria deve ser uma cadeia de linhas, polígono, cadeia de linhas múltiplas ou multipolígonos.

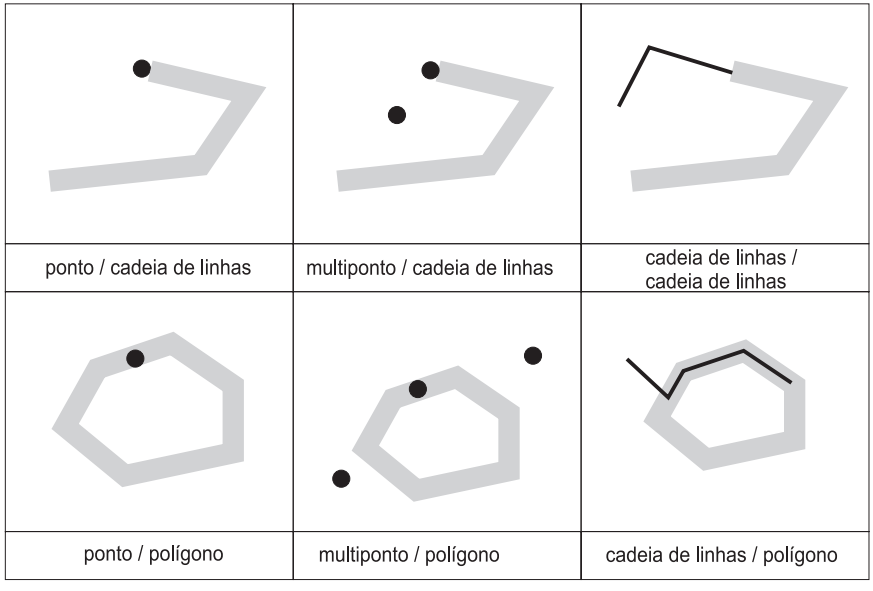


Figura 16. ST_Touches

As matrizes padrão mostram que o predicado ST_Touches retorna VERDADEIRO quando os interiores da geometria não fizerem interseção e o

limite de uma delas fizer interseção com o interior ou limite de outra.

Tabela 47. Matriz para ST_Touches (1)

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	F	T	*
	Limite	*	*	*
	Exterior	*	*	*

Tabela 48. Matriz para ST_Touches (2)

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	F	*	*
	Limite	T	*	*
	Exterior	*	*	*

Tabela 49. Matriz para ST_Touches (3)

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	F	*	*
	Limite	*	T	*
	Exterior	*	*	*

Para obter mais informações, consulte “ST_Touches” na página 276.

ST_Overlaps

ST_Overlaps compara duas geometrias da mesma dimensão. Ela retorna 1 (VERDADEIRO) se o conjunto de interseção resultar numa geometria diferente de ambas, mas que tenha a mesma dimensão.

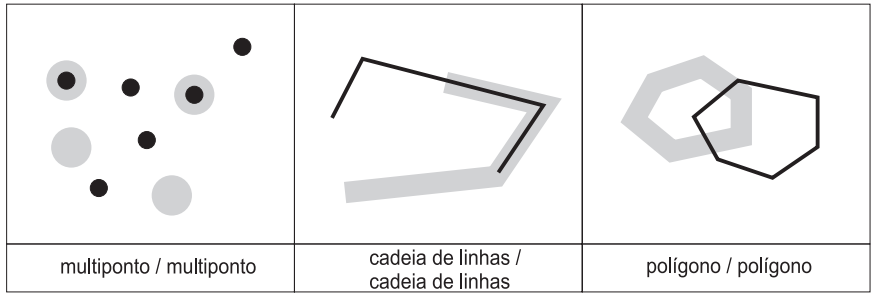


Figura 17. ST_Overlaps

A matriz padrão na Tabela 50 na página 148 aplica-se a sobreposições polígono / polígono, multipontos / multipontos e multipolígonos / multipolígonos. Para estas combinações, o predicado de

sobreposição retornará VERDADEIRO se o interior de ambas geometrias fizerem interseção com o interior e exterior de outras.

Tabela 50. Matriz para ST_Overlaps (1)

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	T	*	T
	Limite	*	*	*
	Exterior	T	*	*

A matriz padrão na Tabela 51 aplica-se a sobreposições de cadeia de linhas / cadeia de linhas e cadeia de linhas múltiplas / cadeia de linhas múltiplas. Neste caso, a interseção das geometrias deverá resultar numa geometria que tenha uma dimensão 1 (outra cadeia de linha). Se a dimensão da interseção dos interiores for 1, o predicado ST_Overlaps retornaria FALSO, no entanto, o predicado ST_Crosses retornaria VERDADEIRO.

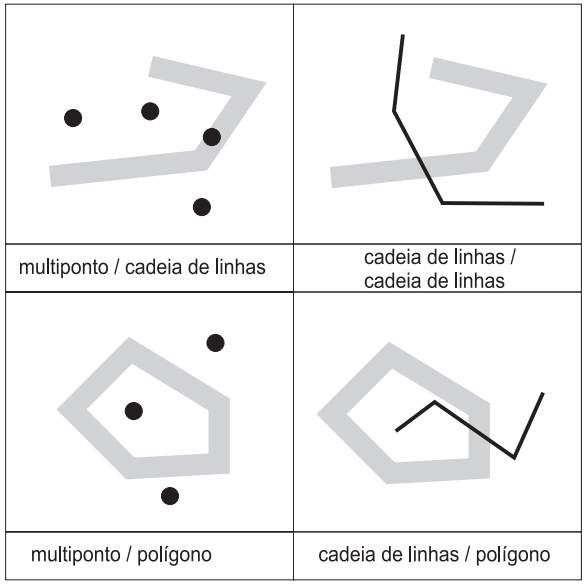
Tabela 51. Matriz para ST_Overlaps (2)

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	1	*	T
	Limite	*	*	*
	Exterior	T	*	*

Para obter mais informações, consulte “ST_Overlaps” na página 257.

ST_Crosses

ST_Crosses retorna 1 (VERDADEIRO) se a interseção resultar numa geometria cuja dimensão seja uma menor que a dimensão máxima das duas geometrias de origem e o conjunto de interseção seja interior em ambas. ST_Crosses retorna 1 (VERDADEIRO) somente para comparações de multipontos / polígono, multipontos / cadeia de linha, cadeia de linha / cadeia de linha, cadeia de linha / polígono e cadeia de linha / multipolígonos.



A matriz padrão na Tabela 52 aplica-se a multipontos / cadeia de linhas, multipontos / cadeia de linhas múltiplas, multipontos / polígono, multipontos / multipolígonos, cadeia de linhas / polígono, cadeia de linhas / multipolígonos. A matriz indica que os interiores devem fazer interseção e que o interior da primária (geometria *a*) deve fazer interseção com o interior da secundária (geometria *b*).

Tabela 52. Matriz para ST_Crosses (1)

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	T	*	T
	Limite	*	*	*
	Exterior	*	*	*

A matriz padrão na Tabela 53 na página 150 aplica-se à cadeia de linhas / cadeia de linhas, cadeia de linhas / cadeia de linhas múltiplas e cadeia de linhas múltiplas / cadeia de linhas múltiplas. A matriz indica que a dimensão da interseção dos interiores deve ser 0 (interseção num ponto). Se a dimensão desta interseção for 1 (interseção numa cadeia de linha), o predicado ST_Crosses retornará FALSO; no entanto, o predicado ST_Overlaps retornará VERDADEIRO.

Tabela 53. Matriz para ST_Crosses (2)

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	0	*	*
	Limite	*	*	*
	Exterior	*	*	*

Para obter mais informações, consulte “ST_Crosses” na página 203.

ST_Within

ST_Within retorna 1 (VERDADEIRO) se a primeira geometria estiver completamente dentro da segunda. ST_Within retorna o resultado oposto exato de ST_Contains.

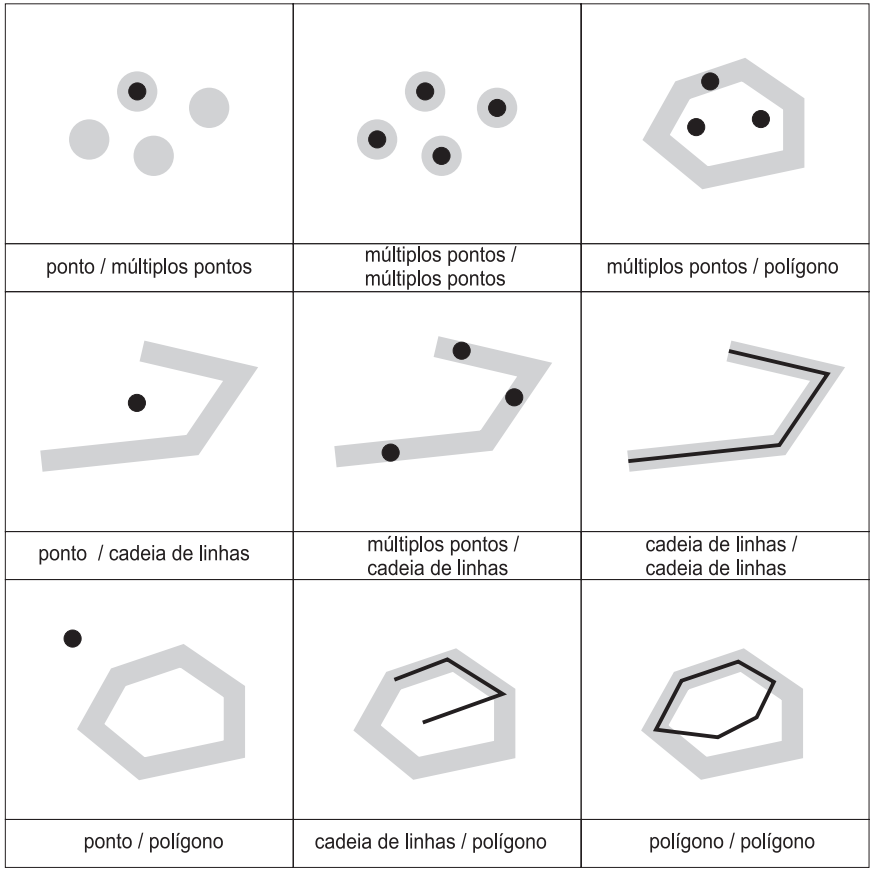


Figura 18. Within

A matriz do predicado ST_Within indica que os interiores das duas geometrias devem fazer interseção e que o interior e o limite da geometria primária (geometria *a*) não deve fazer interseção com o exterior da segunda (geometria *b*).

Tabela 54. Matriz para ST_Within

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	T	*	F
	Limite	*	*	F
	Exterior	*	*	*

Para obter mais informações, consulte “ST_Within” na página 279.

ST_Contains

ST_Contains retorna 1 (VERDADEIRO) se a segunda geometria estiver completamente dentro da primeira. O predicado ST_Contains retorna o resultado oposto exato do predicado ST_Within.

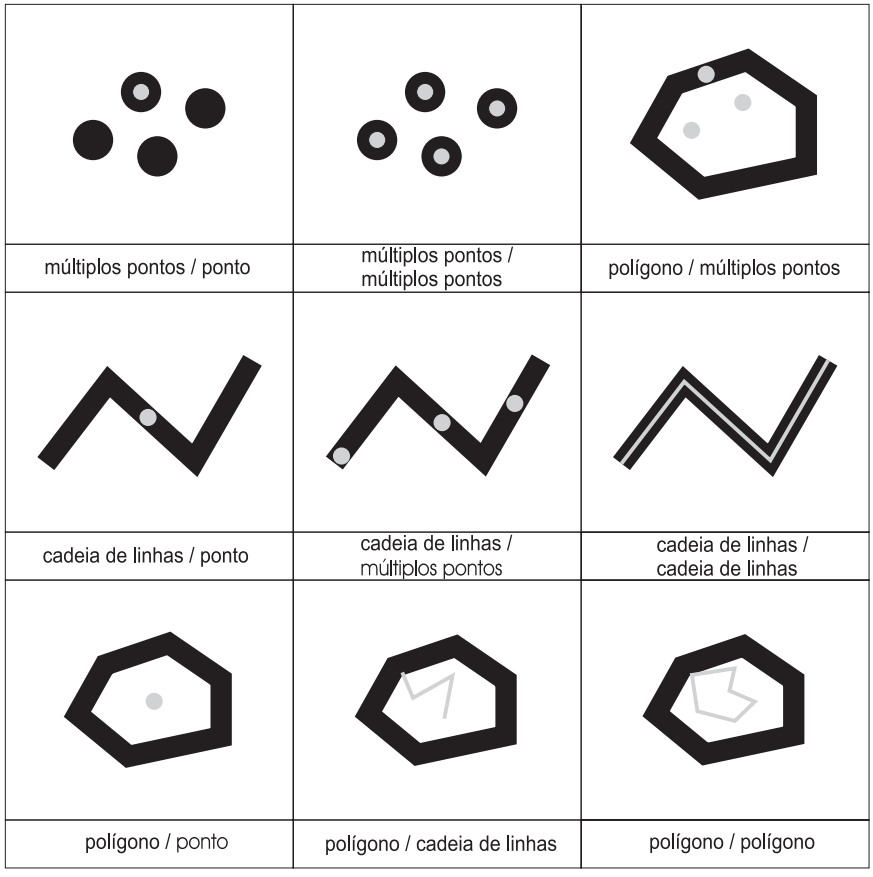


Figura 19. ST_Contains

A matriz padrão do predicado ST_Contains indica que o interior das duas geometrias deve fazer interseção e que o interior e limite da secundária (geometria *b*) não deve fazer interseção com o exterior da primária (geometria *a*).

Tabela 55. Matriz para ST_Contains

		b		
		Interior	**Limite**	**Exterior**
a	**Interior**	T	*	*
	Limite	*	*	*
	Exterior	F	F	*

Para obter mais informações, consulte “ST_Contains” na página 197.

ST_Relate

A função ST_Relate compara duas geometrias e retornará 1 (VERDADEIRO) se as geometrias atenderem às condições especificadas pela cadeia de matrizes padrão DE-91M, do contrário, a função retornará 0 (FALSO). Para obter mais informações, consulte “ST_Relate” na página 270.

ST_Distance

A função ST_Distance informa a distância mínima que separa dois recursos desunidos. Se os recurso não estiverem desunidos, a função irá informar uma distância mínima de O.

Por exemplo, ST_Distance poderia informar a menor distância entre dois locais que uma aeronave deveria viajar. A Figura 20 ilustra estas informações.

Figura 20. Distância mínima entre duas cidades. ST_Distance pode tomar as coordenadas para as cidades de Los Angeles e Chicago como entrada e retornar uma valor indicando a distância mínima entre estes locais.

Para obter mais informações, consulte “ST_Distance” na página 210.

Funções que geram novas geometrias a partir de existentes

O DB2 Spatial Extender fornece predicados e funções de transformação que geram novas geometrias a partir de existentes.

ST_Intersection

A função ST_Intersection retorna o conjunto de interseção de duas geometrias. Ele sempre retorna como uma coleção que indica a dimensão mínima das geometrias de origem. Por exemplo, para uma cadeia de linhas que faça interseção com um polígono, a função de interseção retorna uma cadeia de linhas múltiplas composta dessa parte da cadeia de linhas comum ao interior e ao limite do polígono. A cadeia de linhas múltiplas conterá mais de uma cadeia de linhas se a cadeia de linhas de origem fizer interseção com o polígono com dois ou mais segmentos descontínuos. Se as geometrias não

fizerem interseção ou se a interseção resultar numa dimensão menor que ambas as geometrias de origem, será retornada uma geometria vazia.

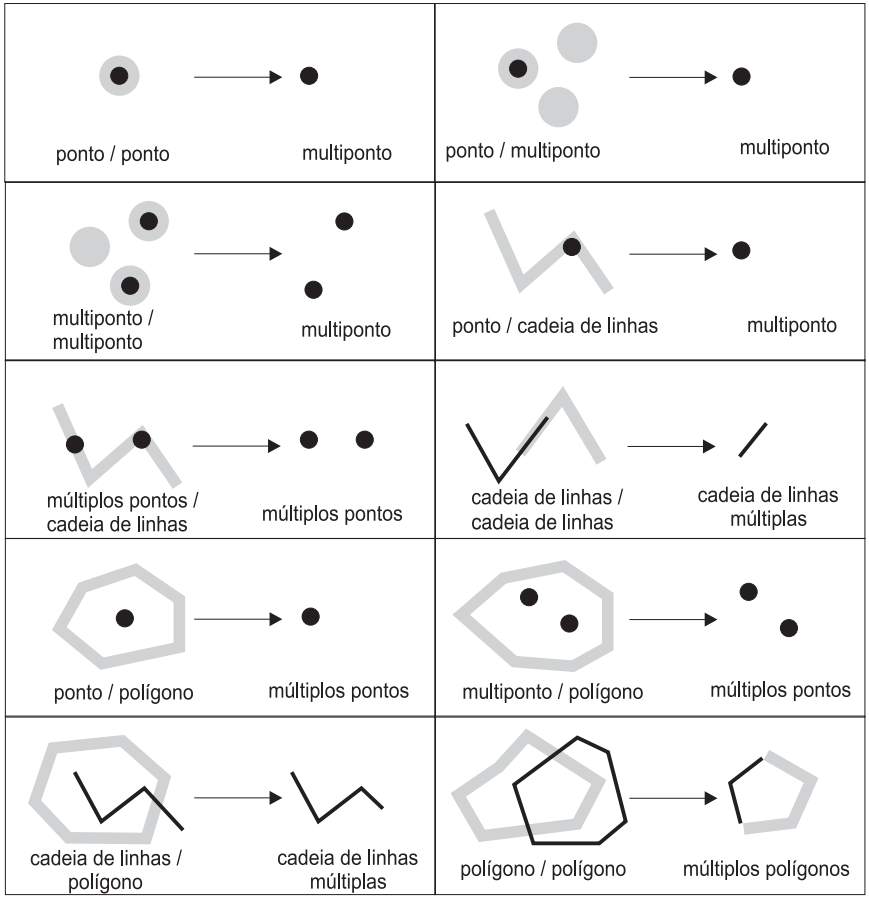


Figura 21. ST_Intersection. Exemplos da função ST_Intersection.

Para obter mais informações, consulte “ST_Intersection” na página 229.

ST_Difference

A função ST_Difference retorna a parte da geometria primária que não fez interseção com a segunda geometria. Este é o E NÃO (AND NOT) lógico de espaço. A função ST_Difference opera somente em geometrias de dimensão semelhante e retorna uma coleção que tem a mesma dimensão que as geometrias de origem. No caso de as geometrias serem iguais, será retornado uma geometria vazia.

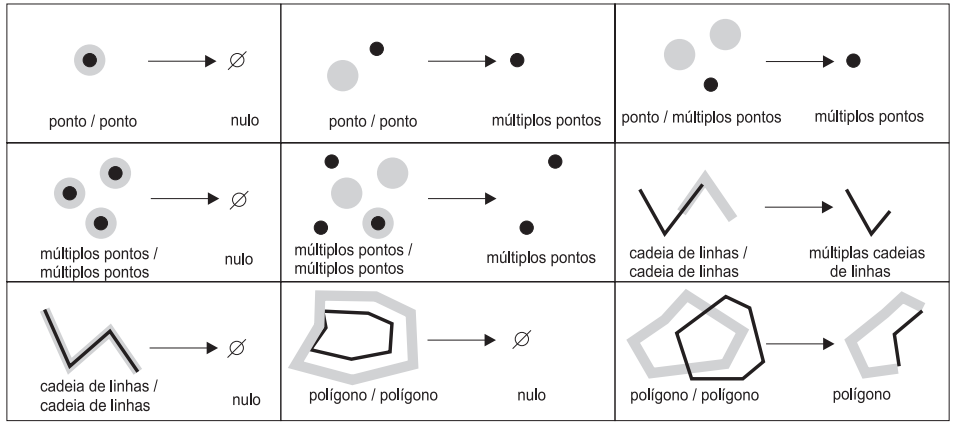


Figura 22. ST_Difference

Para obter mais informações, consulte “ST_Difference” na página 205.

ST_Union

A função ST_Union retorna o conjunto de união de duas geometrias. Este é o OU (OR) lógico de espaço. As geometrias de origem deve ser de dimensão semelhante. ST_Union sempre retorna o resultado como uma coleção.

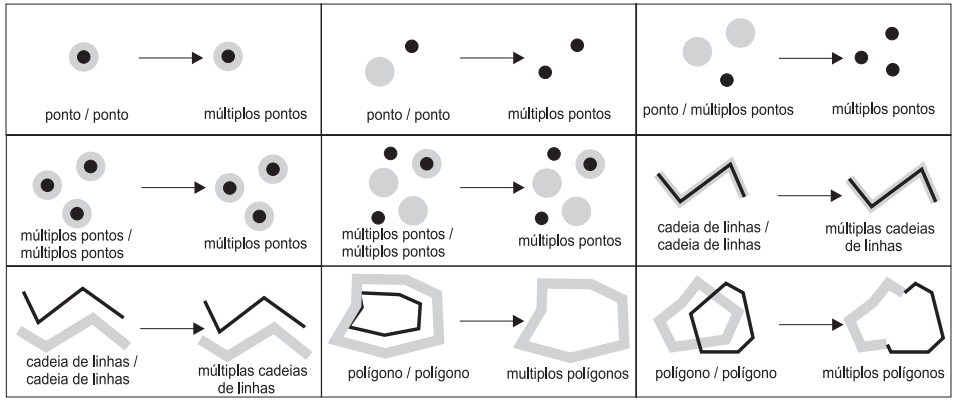


Figura 23. ST_Union

Para obter mais informações, consulte “ST_Union” na página 278.

ST_Buffer

A função ST_Buffer gera uma geometria englobando uma geometria numa distância especificada. Um polígono resultará quando uma geometria primária estiver em buffer, sempre que os elementos de uma coleção estiverem próximos o suficiente, de modo que todos os polígonos do buffer estejam sobrepostos. No entanto, quando há separação suficiente entre os elementos

de uma coleção em buffer, resultarão polígonos individuais do buffer indicando, nesse caso, que a função ST_Buffer retorna multipolígonos.

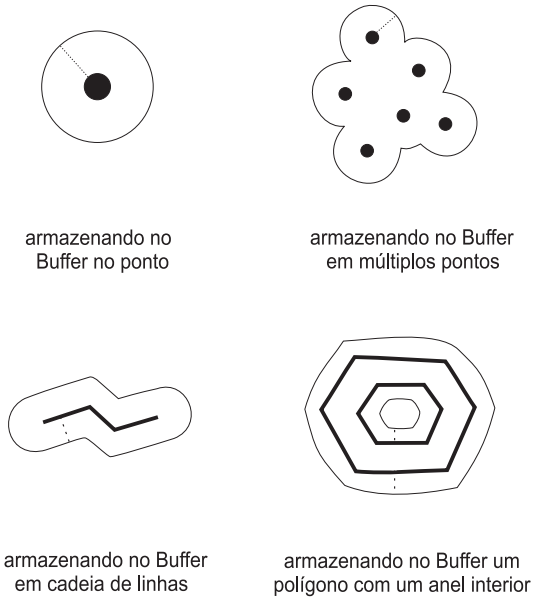


Figura 24. ST_Buffer

A função ST_Buffer aceita distâncias positivas e negativas, no entanto, somente geometrias com dimensão dois (polígonos e multipolígonos) aplicam um buffer negativo. O valor absoluto da distância do buffer é usado sempre que a dimensão da geometria de origem for menor que 2 (todas as geometrias que não sejam polígonos ou multipolígonos).

Em geral, para anéis externos, as distâncias do buffer positivo geram anéis de polígonos que estejam distantes do centro da geometria de origem; distâncias de buffer negativo geram anéis de polígonos ou multipolígonos em direção ao centro. Para anéis internos de um polígono ou multipolígonos, a distância de um buffer positivo gera um anel de buffer em direção ao centro e uma distância de um buffer negativo gera um anel de buffer a partir do centro.

O processo de buffer combina polígonos que se sobrepõem. As distâncias negativas maiores que metade da largura máxima do interior de um polígono resultam numa geometria vazia.

Para obter mais informações, consulte “ST_Buffer” na página 194.

LocateAlong

Para geometrias que contêm medidas, a localização de uma determinada medida pode ser encontrada com a função LocateAlong. LocateAlong retorna a localização como multipontos. Se a dimensão da geometria de origem for zero 0 (por exemplo, um ponto e multipontos), será solicitada uma correspondência exata e esses pontos que têm valor de medida correspondente retornarão como multipontos. Contudo, para geometrias de origem cuja dimensão seja maior que 0, a localização será interpolada. Por exemplo, se o valor de medida digitado for 5,5 e as medidas nos vértices de uma cadeia de linha forem 3, 4, 5, 6 e 7 respectivamente, o ponto interpolado que ficar exatamente entre os vértices com valores de medida 5 e 6 retornarão.

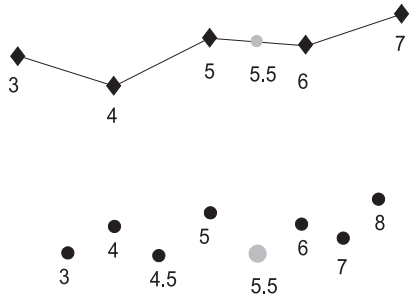


Figura 25. LocateAlong

Para obter mais informações, consulte “LocateAlong” na página 173.

LocateBetween

A função LocateBetween retorna o conjunto de caminhos ou localizações que ficam entre dois valores de medida de uma geometria de origem que tenha medidas. Se a dimensão de geometria de origem for 0, LocateBetween retornará multipontos contendo todos os pontos cujas medidas estejam entre as duas medidas de origem. Para geometrias de origem cuja dimensão seja maior que 0, LocateBetween retornará uma cadeia de linhas múltiplas se for possível interpolar um caminho; do contrário, LocateBetween retornará multipontos contendo as localizações do ponto. Um ponto vazio retornará sempre que LocateBetween não conseguir interpolar um caminho ou encontrar uma localização entre as medidas. LocateBetween executa uma pesquisa inclusiva das geometrias; portanto as medidas das geometrias devem ser maiores ou iguais à medida de *origem* e menores ou iguais à medida de *destino*.

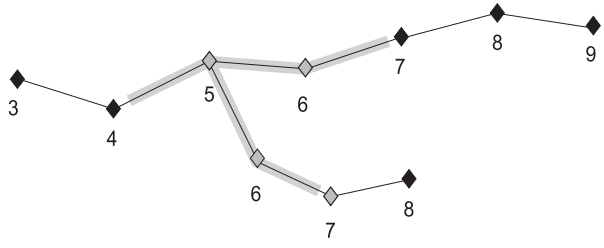


Figura 26. LocateBetween

Para obter mais informações, consulte “LocateBetween” na página 175.

ST_ConvexHull
A função ST_ConvexHull retorna o polígono de casca convexa de qualquer geometria que tenha pelo menos três vértices formando um convexo. Se os vértices da geometria não formarem um convexo, ST_ConvexHull retornará um nulo. ST_ConvexHull geralmente é a primeira etapa em mosaico utilizada para criar uma rede TIN a partir de um conjunto de pontos.

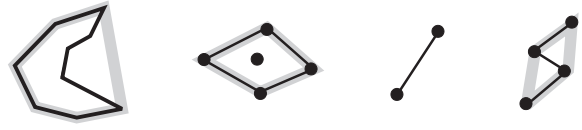


Figura 27. ST_ConvexHull

Para obter mais informações, consulte “ST_ConvexHull” na página 199.

ST_Polygon
Gera um polígono a partir de uma cadeia de linha. Para obter mais informações, consulte “ST_Polygon” na página 269.

Funções que convertem o formato dos valores de uma geometria

O DB2 Spatial Extender suporta três formatos de troca de dados GIS:

- Representação de texto reconhecida
- Representação do modo binário reconhecida
- Representação de shape do modo binário ESRI

Representação de texto reconhecida

O DB2 Spatial Extender possui várias funções que geram geometrias a partir de descrições de textos.

ST_WKTToSQL
Cria uma geometria a partir da representação de texto de qualquer

tipo de geometria. Nenhum identificador do sistema de referência espacial deverá ser especificado. Para obter mais informações, consulte “ST_WKTToSQL” na página 282.

ST_GeomFromText
Cria uma geometria a partir da representação de texto de qualquer tipo de geometria. Nenhum identificador do sistema de referência espacial deverá ser especificado. Para obter mais informações, consulte “ST_GeometryFromText” na página 217.

ST_PointFromText
Cria um ponto a partir de uma representação de texto de ponto. Para obter mais informações, consulte “ST_PointFromText” na página 260.

ST_LineFromText
Cria uma cadeia de linhas a partir de uma representação de texto da cadeia de linhas. Para obter mais informações, consulte “ST_LineFromText” na página 243.

ST_PolyFromText
Cria um polígono a partir de uma representação do texto do polígono. Para obter mais informações, consulte “ST_PolyFromText” na página 266.

ST_MPointFromText
Cria multipontos a partir de uma representação de multipontos. Para obter mais informações, consulte “ST_MPointFromText” na página 249.

ST_MLineFromText
Cria uma cadeia de linhas múltiplas a partir de uma representação da cadeia de linhas múltiplas. Para obter mais informações, consulte “ST_MLineFromText” na página 246.

ST_MPolyFromText
Cria um multipolígono a partir de uma representação de multipolígono. Para obter mais informações, consulte “ST_MPolyFromText” na página 251.

A representação de texto é uma cadeia ASCII. Ela permite que a geometria seja trocada no formato de texto ASCII. Estas funções não exigem a definição de estruturas de programas especiais para mapear uma representação do modo binário. Portanto, podem ser usadas num programa 3GL ou 4GL.

A função ST_AsText converte o valor de uma geometria existente em representação de texto. Para obter mais informações, consulte “ST_AsText” na página 191.

Para obter uma descrição detalhada de representações de texto reconhecidas, consulte “As representações de texto reconhecidas de OGC” na página 297.

Representação do modo binário reconhecida

O DB2 Spatial Extender possui várias funções que geram geometrias a partir de representações do modo binário reconhecidas (WKB).

ST_WKBToSQL
Cria uma geometria a partir de uma representação do modo binário reconhecida de qualquer tipo de geometria. Nenhum identificador do sistema de referência espacial deverá ser especificado. Para obter mais informações, consulte “ST_WKBToSQL” na página 280.

ST_GeomFromWKB
Cria uma geometria a partir de uma representação do modo binário reconhecida de qualquer tipo de geometria. Nenhum identificador do sistema de referência espacial deverá ser especificado. Para obter mais informações, consulte “ST_GeomFromWKB” na página 219.

ST_PointFromWKB
Cria um ponto a partir de uma representação do modo binário reconhecida de um ponto. Para obter mais informações, consulte “ST_PointFromWKB” na página 261.

ST_LineFromWKB
Cria uma cadeia de linha a partir de uma representação do modo binário reconhecida de uma cadeia de linha. Para obter mais informações, consulte “ST_LineFromWKB” na página 244.

ST_PolyFromWKB
Cria um polígono a partir de uma representação do modo binário reconhecida de um polígono. Para obter mais informações, consulte “ST_PolyFromWKB” na página 267.

ST_MPointFromWKB
Cria multipontos a partir de uma representação do modo binário reconhecida de multipontos. Para obter mais informações, consulte “ST_MPointFromWKB” na página 250.

ST_MLineFromWKB
Cria uma cadeia de linhas múltiplas a partir de uma representação do modo binário reconhecida de uma cadeia de linhas múltiplas.Para obter mais informações, consulte “ST_MLineFromWKB” na página 247.

ST_MPolyFromWKB
Cria um multipolígono a partir de uma representação do modo binário reconhecida de um multipolígono. Para obter mais informações, consulte “ST_MPolyFromWKB” na página 252.

A representação do modo binário reconhecida é um fluxo contíguo de bytes. Ela permite que a geometria sejam trocada entre um cliente ODBC e um banco de dados SQL no modo binário. Estas funções de geometria exigem a

definição das estruturas C para mapear a representação do modo binário. Portanto, elas destinam-se ao uso dentro de um programa 3GL e não se adequam a um ambiente 4GL.

A função ST_AsBinary converte um valor de geometria existente em representação do modo binário reconhecida. Para obter mais informações, consulte “ST_AsBinary” na página 190.

Para obter uma descrição detalhada de representações do modo binário reconhecidas, consulte “As representações (WKB) do modo binário reconhecidas” na página 302.

Representação de shape ESRI

O DB2 Spatial Extender possui várias funções que geram geometrias a partir de uma representação de shape ESRI. A representação de shape ESRI suporta coordenadas Z e medidas, além das representações bidimensionais suportadas pelas representações de texto e do modo binário reconhecidas.

ShapeToSQL
Cria uma geometria a partir de um shape de qualquer tipo de geometria. Nenhum identificador do sistema de referência espacial deverá ser especificado. Para obter mais informações, consulte “ShapeToSQL” na página 186.

GeometryFromShape
Cria uma geometria a partir de um shape de qualquer tipo de geometria. Nenhum identificador do sistema de referência espacial deverá ser especificado. Para obter mais informações, consulte “GeometryFromShape” na página 165.

PointFromShape
Cria um ponto a partir de um shape. Para obter mais informações, consulte “PointFromShape” na página 182.

LineFromShape
Cria uma cadeia de linhas a partir de um shape polilinha. Para obter mais informações, consulte “LineFromShape” na página 171.

PolyFromShape
Cria um polígono a partir de um shape de polilinha. Para obter mais informações, consulte “PolyFromShape” na página 184.

MPointFromShape
Cria múltiplos pontos a partir de um shape de multipontos. Para obter mais informações, consulte “MPointFromShape” na página 180.

MLineFromShape
Cria uma cadeia de linhas múltiplas a partir de um shape de polilinha de várias partes. Para obter mais informações, consulte “MLine FromShape” na página 178.

MPolyFromShape
Cria um multipolígono a partir de um shape de polígono de multipartes. Para obter mais informações, consulte "MPolyFromShape" na página 181.

A sintaxe geral destas funções é a mesma. O primeiro argumento é a representação de shape fornecida como tipo de dados BLOB. O segundo argumento é o identificador de referência espacial que será atribuído à geometria. Por exemplo, a função GeometryFromShape apresenta a seguinte sintaxe:

```
GeometryFromShape(shapegeometry, SRID)
```

Para mapear a representação do modo binário, estas funções de shape exigem a definição de estruturas C. Portanto, elas destinam-se ao uso dentro de um programa 3GL e não se adequam a um ambiente 4GL.

A função AsBinaryShape converte o valor de uma geometria em uma representação do shape ESRI. Para obter mais informações, consulte "AsBinaryShape" na página 164.

Para obter uma descrição detalhada de representações de shape, consulte "As representações de shape ESRI" na página 306.

Capítulo 14. Funções espaciais para consultas SQL

Este capítulo relaciona as funções disponíveis que você pode chamar quando consultar os dados especiais. Cada uma das funções é descrita em uma seção que mostra a sintaxe, o tipo de retorno e os exemplos de código. Alguns dos exemplos deste capítulo incluem uma instrução CREATE TABLE em que uma ou mais colunas são definidas como colunas espaciais.

As considerações a seguir se aplicam às funções espaciais:

- Os exemplos deste capítulo estão qualificados com o nome da biblioteca db2gse. Em vez de qualificar explicitamente cada uma das funções espaciais e tipo com db2gse, você pode definir o caminho da função para que inclua o db2gse.
- Para poder inserir os dados em uma coluna espacial:
  - Talvez você precise aumentar o parâmetro udf_mem_sz. A definição inicial sugerida é 2048. Se o valor 2048 for inadequado, aumente o parâmetro udf_mem_sz em quantidades de 256.
  - É preciso registrá-lo como uma camada. Para obter mais informações sobre o registro de uma coluna espacial como uma camada, consulte “Capítulo 4. Definindo colunas espaciais, registrando-as como camadas e ativando um geocoder para mantê-las” na página 33.

AsBinaryShape

O AsBinaryShape pega um objeto de geometria e apresenta um BLOB.

Sintaxe

db2gse.AsBinaryShape(g db2gse.ST_Geometry)

Tipo de retorno

BLOB(1m)

Exemplos

O fragmento de código a seguir demonstra como a função do AsBinaryShape converte os polígonos de zona da tabela SENSITIVE_AREAS em polígonos shape. Tais polígonos shape são passados para a função de desenho do polígono (draw_polygon) exibição.

```
/* Criar a expressão SQL. */
strcpy(sqlstmt, "select db2gse.AsBinaryShape (zone) from SENSITIVE_AREAS
where db2gse.EnvelopesIntersect(zone, db2gse.PolyFromShape(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Preparar as instruções SQL. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* Definir o comprimento de pcbvalue1 do shape. */
pcbvalue1 = blob_len;

/* Fazer o bind do parâmetro do shape */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB, blob_len,
0, shape, blob_len, &pcbvalue1);

/* Executar a consulta */
rc = SQLExecute (hstmt);

/* Atribuir os resultados da consulta (os polígonos da Zone) a
   fetched_binary variable. */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* Buscar cada polígono dentro da janela de exibição e exibi-lo. */

while(SQL_SUCCESS == (rc = SQLFetch(hstmt)))
  draw_polygon(fetched_binary);
```

GeometryFromShape

O GeometryFromShape pega um shape e uma identidade do sistema de referência espacial para retornar um objeto de geometria.

Sintaxe

db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

O seguinte fragmento de código C contém funções ODBC incorporadas às funções SQL do DB2 Spatial Extender que inserem dados na tabela LOTS.

A tabela LOTS foi criada com duas colunas: a coluna LOT_ID, que identifica cada lote exclusivamente e a coluna polígomo do LOT, que contém a geometria de cada lote.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot     db2gse.ST_MultiPolygon);
```

A função GeometryFromShape converte shapes em geometria do DB2 Spatial Extender. A instrução INSERT inteira é copiada em shp_sql. A instrução INSERT contém marcadores do parâmetro para a ceitar os dados do LOT_ID e os dados do LOT dinamicamente.

```
/* Criar a instrução insert do SQL para preencher a id do lote e o
  multipolígono do lote. Os pontos de interrogação são marcadores
  do parâmetro que
 indicam a id do_lote e os valores do lote
  que serão recuperados no
 tempo de execução.*/
strcpy (shp_sql,"insert into LOTS (lot_id, lot) values (?,
db2gse.GeometryFromShape (cast(? as blob(1m)), db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da chave ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Fazer o bind do shape ao segundo parâmetro. */
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
        SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

    /* Executar a instrução insert . */
    rc = SQLExecute (hstmt);
```

EnvelopesIntersect

EnvelopesIntersect retorna 1 (VERDADEIRO) se os envelopes de duas geometrias tiverem interseção; caso contrário retonará 0 (FALSO).

Sintaxe

db2gse.EnvelopesIntersect(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A função da janela get_ recupera as coordenadas da janela de exibição da aplicação. O parâmetro da janela é, na verdade, uma estrutura de shape do polígono, que contém uma cadeia de coordenadas que representam o polígono de exibição. A função PolyFromShape converte o formato da janela de exibição em um polígono do DB2 Spatial Extender que a função EnvelopesIntersect utiliza como seu envelope de interseção. Todos os polígonos da zona SENSITIVE_SAREAS que fazem interseção com o interior ou com o limite da janela de exibição são retornados. Cada polígono é obtido do conjunto de resultados e passado à função draw_polygon.

```
/* Obter as coordenadas da janela de exibição como um shape do polígono.
get_window(&window)

/* Criar a expressão SQL. A função db2gse.EnvelopesIntersect
  será usada para limitar o conjunto de resultados somente nesses
  polígonos da zona
  que fazem interseção com o envelope da janela de exibição. */
strcpy(sqlstmt, "select db2gse.AsBinaryShape(zone) from SENSITIVE_AREAS where
db2gse.EnvelopesIntersect (zone, db2gse.PolyFromShape(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Definir blob_len para o comprimento do byte de um polígono
   de shape de 5 pontos. */
blob_len = 128;

/* Preparar as instruções SQL. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* Definir pcbvalue1 para o shape da janela */
pcbvalue1 = blob_len;

/* Fazer o bind do parâmetro do shape */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB,
blob_len,0, window, blob_len, &pcbvalue1);

/* Executar a consulta */
rc = SQLExecute (hstmt);

/* Atribuir os resultados da consulta, (os polígonos da Zona) para a
   fetched_binary variable. */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);
```

```
/* Buscar cada polígono dentro da janela de exibição e exibi-lo. */
while(SQL_SUCCESS == (rc = SQLFetch(hstmt))
  draw_polygon(fetched_binary);
```

Is3d

Is3d toma um objeto de geometria e retorna 1 (VERDADEIRO) se o objeto possuir coordenadas 3D; caso contrário, retornará 0 (FALSO).

Sintaxe

db2gse.Is3d(g db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A seguinte instrução CREATE TABLE cria a tabela THREED_TEST, a qual possui duas colunas: a coluna GID do tipo inteiro e a coluna de geometria G1.

```
CREATE TABLE THREED_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

As instruções INSERT inserem dois pontos na tabela THREED_TEST. O primeiro ponto não contém coordenadas Z, ao passo que o segundo sim.

```
INSERT INTO THREED_TEST
VALUES(1, db2gse.ST_PointFromText('point (10 10)', db2gse.coordref()..srid(0)))

INSERT INTO THREED_TEST
VALUES (2, db2gse.ST_PointFromText('point z (10.92 10.12 5)',
db2gse.coordref()..srid(0)))
```

A instrução SELECT a seguir relaciona o conteúdo da coluna GID aos resultados da função Is3d. A função retorna 0 para a primeira linha, o qual não possui coordenadas Z e 1 para a segunda linha, que possui coordenadas Z.

```
SELECT gid, db2gse.Is3d (g1) "Is it 3d?" FROM THREED_TEST
```

O seguinte conjunto de resultados retorna.

```
gid      Is it 3d?
------   ----------
     1            0
     2            1
```

IsMeasured

IsMeasured toma um objeto de geometria e retorna 1 (VERDADEIRO) se o objeto possuir medidas; caso contrário, retornará 0 (FALSO).

Sintaxe

db2gse.IsMeasured(g db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A seguinte instrução CREATE TABLE cria a tabela MEASURE_TEST, a qual possui duas colunas. A coluna GID identifica exclusivamente as linhas e a coluna G1 armazena as geometrias do ponto.

```
CREATE TABLE MEASURE_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

A seguinte instrução INSERT insere dois registros na tabela MEASURE_TEST. O primeiro registro armazena um ponto que não possui uma medida. O segundo ponto do registro não possui uma medida.

```
INSERT INTO MEASURE_TEST
VALUES(1, db2gse.ST_PointFromText('point (10 10)', db2gse.coordref()..srid(0)))

INSERT INTO MEASURE_TEST
VALUES (2, db2gse.ST_PointFromText('point m (10.92 10.12 5)',

db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostram a coluna GID com os resultados da função IsMeasured. A função IsMeasured retorna 0 para a primeira linha porque o ponto não contém uma medida. Ela retorna 1 para a segunda linha porque o ponto tem medidas.

```
SELECT gid, db2gse.IsMeasured (g1) "Has measures?" FROM MEASURE_TEST
gid      Contém medidas
------   ----------
     1            0
     2            1
```

LineFromShape

LineFromShape toma um shape do tipo ponto e uma identidade do sistema de referência espacial e retorna uma cadeia de linhas.

Sintaxe

db2gse.Line FromShape(ShapeLineString Blob(1M), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_LineString

Exemplos

O seguinte fragmento de código preenche a tabela SEWERLINES com id, classe de tamanho e geomeria exclusivos de cada linha sewer.

A instrução CREATE TABLE cria a tabela SEWERLINES, a qual possui três colunas. A primeira coluna, SEWER_ID, identifica exclusivamente cada linha do sewer. A segunda coluna, CLASS, do tipo inteiro identifica o tipo de linha sewer, que geralmente está associada a uma capacidade da linha. A terceira coluna, SEWER, do tipo cadeia de linhas armazena a geometria da linha sewer.

```
CREATE TABLE SEWERLINES (sewer_id   integer, class  integer,
                         sewer db2gse.ST_LineString);

/* Criar a instrução insert do SQL para preencher a id do sewer, o tamanho e
   a cadeia de linha sewer. Os pontos de interrogação são marcadores de
   parâmetros que
   indicam a id do sewer, a classe e os valores de geometria do sewer
   que serão
   recuperados no tempo de execução. */
strcpy (shp_sql,"insert into sewerlines (sewer_id,class,sewer)
values (?,?, db2gse.Line FromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da chave ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_id, 0, &pcbvalue1);

/* Fazer o bind do valor da classe de inteiro ao segundo parâmetro. */
pcbvalue2 = 0;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_class, 0, &pcbvalue2);;

/* Fazer o bind do shape ao terceiro parâmetro. */
```

```
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, sewer_shape, blob_len, &pcbvalue3);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

LocateAlong

LocateAlong toma um objeto de geometria e uma medida para retornar como um multiponto do conjunto de pontos encontrados na medida.

Sintaxe

db2gse.LocateAlong(g db2gse.ST_Geometry, adistance Double)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

A seguinte instrução CREATE TABLE cria a tabela LOCATEALONG_TEST. LOCATEALONG_TEST apresenta duas colunas: a coluna GID, que identifica exclusivamente cada linha e a coluna da geometria G1, que armazena a geometria de amostra.

```
CREATE TABLE LOCATEALONG_TEST (gid integer, g1 db2gse.ST_Geometry)
```

A seguinte instrução INSERT insere duas linhas. A primeira é uma cadeia de linhas múltiplas; a segunda é de multipontos.

```
INSERT INTO db2gse.LOCATEALONG_TEST VALUES(
1, db2gse.ST_MLineFromText('multilinestring m ((10.29 19.23 5,23.82 20.29 6,
                                        30.19 18.47 7,
45.98 20.74 8), (23.82 20.29 6,30.98 23.98 7,42.92 25.98 8))',
                                db2gse.coordref()..srid(0)))

INSERT INTO db2gse.LocateAlong_TEST VALUES(
2, db2gse.ST_MPointFromText('multipoint m (10.29 19.23 5,23.82 20.29 6,
 30.19 18.47 7,45.98 20.74 8,23.82 20.29 6,30.98 23.98 7,42.92 25.98)',

db2gse.coordref()..srid(0)))
```

Na seguinte instrução SELECT e no conjunto de resultados correspondente, a função LocateAlong está direcionada para localizar pontos cuja medida seja 6,5. A primeira linha retorna um multiponto que contém dois pontos. No entanto, a segunda linha retornou um ponto vazio. Para recursos lineares (geometria com dimensão maior que 0), LocateAlong pode interpolar o ponto, no entanto, para multipontos, a medida de destino deve corresponder exatamente.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.LocateAlong (g1,6.5)) AS
varchar(96))
"Geometry"
FROM LOCATEALONG_TEST

GID         Geometry
----------- ------------------------------------------------------------

          1 MULTIPOINT M ( 27.01000000 19.38000000 6.50000000, 27.40000000
```

```
22.14000000 6.50000000)
          2 POINT EMPTY

  2 record(s) selected.
```

Na seguinte instrução SELECT e no conjunto de resultados correspondente, a função LocateAlong retorna multipontos para ambas as linhas. A medida de destino de 7 corresponde às medidas tanto nos dados de origem da cadeia de linhas múltiplas como de multipontos.

```
SELECT gid,CAST(db2gse.ST_AsText(db2gse.LocateAlong (g1,7)) AS varchar(96))
   "Geometry"
FROM LOCATEALONG_TEST

GID         Geometry

----------- ----------------------------------------------------------------
          1 MULTIPOINT M ( 30.19000000 18.47000000 7.00000000, 30.98000000
 23.98000000 7.00000000)
          2 MULTIPOINT M ( 30.19000000 18.47000000 7.00000000, 30.98000000
 23.98000000 7.00000000)

  2 record(s) selected.
```

LocateBetween

O LocateBetween pega um objeto de geometria e duas localizações de medida e retorna uma geometria que representa o conjunto de caminhos desconectados entre as duas localizações de medida.

Sintaxe

db2gse.LocateBetween(g db2gse.ST_Geometry, adistance Double, anotherdistance Double)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

A seguinte instrução CREATE TABLE cria a tabela LOCATEBETWEEN_TEST. LOCATEBETWEEN_TEST apresenta duas colunas: a coluna GID, que identifica exclusivamente cada linha e a coluna G1 de cadeia de multilinha, que armazena a geometria de amostra.

```
CREATE TABLE LOCATEBETWEEN_TEST (gid integer, g1 db2gse.ST_Geometry)
```

A seguinte instrução INSERT insere duas linhas na tabela LOCATEBETWEEN_TEST. A primeira linha é uma cadeia de linhas múltiplas e a segunda é um multiponto.

```
INSERT INTO db2gse.LOCATEBETWEEN_TEST
VALUES(1,db2gse.ST_MLineFromText('multilinestring m ((10.29 19.23 5,
                          23.82 20.29 6, 30.19 18.47 7,45.98 20.74 8),
                                      (23.82 20.29 6,30.98 23.98 7,
42.92 25.98 8))',                    db2gse.coordref()..srid(0)))

INSERT INTO db2gse.LOCATEBETWEEN_TEST
VALUES(2, db2gse.ST_MPointFromText('multipoint m (10.29 19.23 5,23.82 20.29 6,
30.19 18.47 7,45.98 20.74 8,23.82 20.29 6,
30.98 23.98 7,42.92 25.98 8)',
                       db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostram a função LocateBetween localizando medidas entre as 6,5 e 7,5 inclusive. A primeira linha retorna uma cadeia de linhas múltiplas com várias cadeias de linhas. A segunda linha retorna um multiponto porque os dados de origem eram multipontos. Quando os dados de origem têm dimensão de 0 (ponto ou multiponto), é solicitada uma correspondência exata.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.LocateBetween (g1,6.5,7.5))
       AS varchar(96)) "Geometry"
FROM LOCATEBETWEEN_TEST

GID         Geometry
----------- ----------------------------------------------------------------
          1 MULTILINESTRING M ( 27.01000000 19.38000000 6.50000000, 31.19000000
18.47000000 7.00000000,38.09000000 19.61000000 7.50000000),(27.40000000 22.1400
0000 6.50000000, 30.98000000 23.98000000 7.00000000,36.95000000 24.98000000 7.5
```

```
0000000)
          2 MULTIPOINT M ( 30.19000000 18.47000000 7.00000000, 30.98000000 23.9
8000000 7.00000000)

  2 record(s) selected.
```

M

M toma um ponto e retorna sua medida.

Sintaxe

db2gse.M(p db2gse.ST_Point)

Tipo de retorno

Duplo

Exemplos

A seguinte instrução CREATE TABLE cria a tabela M_TEST. M_TEST apresenta duas colunas: a coluna de inteiros GID, que identifica exclusivamente a linha e a coluna de pontos PT1, que armazena a geometria de amostra.

```
CREATE TABLE M_TEST (gid integer, pt1 db2gse.ST_Point)
```

As seguintes instruções INSERT inserem uma linha que contém um ponto com medidas e uma linha que contém um ponto sem medidas.

```
INSERT INTO db2gse.M_TEST
VALUES(1, db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0)))

INSERT INTO db2gse.M_TEST
VALUES(2, db2gse.ST_PointFromText('point  zm(10.02 20.01 5.0 7.0)',
db2gse.coordref()..srid(0)))
```

Na seguinte instrução SELECT e no conjunto de resultados correspondente, a função M lista os valores de medida dos pontos. Como o primeiro ponto não tem medidas, a função M retorna um NULO.

```
SELECT gid, db2gse.M (pt1) "The measure" FROM M_TEST

GID         The measure
----------- ------------------------
          1                         -
          2   +7.00000000000000E+000

  2 record(s) selected.
```

MLine FromShape

MLineFromShape toma um shape do tipo cadeia de linhas múltiplas e uma identidade do sistema de referência e retorna uma cadeia de linhas múltiplas.

Sintaxe

db2gse.MLineFromShape(ShapeMultiLineString Blob(1M), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_MultiLineString

Exemplos

O seguinte fragmento de código preenche a tabela WATERWAYS com uma id exclusiva, um nome e uma cadeia de linhas múltiplas de água.

A tabela WATERWAYS é criada com as colunas ID e NAME que identificam cada sistema de córregos e rios armazenados na tabela. A coluna ÁGUA é uma cadeia de linhas múltiplas pois os sistemas de rios e córregos são geralmente uma agregação de várias cadeias de linhas.

```
CREATE TABLE WATERWAYS (id        integer,
                            name      varchar(128),
                           water     db2gse.ST_MultiLineString);

/* Criar a instrução insert do SQL para preencher a id, nome e
   a cadeia de linhas múltiplas. Os pontos de interrogação são
   marcadores de parâmetros que
   indicam a id, o nome e os valores da água que serão recuperados no
 tempo de execução.*/
strcpy (shp_sql,"insert into WATERWAYS (id,name,water)
values (?,?, db2gse.MLineFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da id ao primeiro parâmetro.*/

pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);
/* Fazer o bind do valor do nome varchar ao segundo parâmetro. */

pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, name_len, 0, &name, name_len, &pcbvalue2);

/* Fazer o bind do shape ao terceiro parâmetro. */
pcbvalue3 = blob_len;
```

```
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, water_shape, blob_len, &pcbvalue3);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

MPointFromShape

MPointFromShape toma um shape do tipo multiponto e uma identidade do sistema de referência espacial para retornar um multiponto.

Sintaxe

db2gse.MPointFromShape(ShapeMultiPoint (1M), srs db2gse.coordref)

Tipo de retorno

db2gse.ST_MultiPoint

Exemplos

Este fragmento de código preenche a tabela SPECIES_SITINGS de um biólogo.

A tabela SPECIES_SITINGS foi criada com três colunas. As colunas espécies e gêneros identificam exclusivamente cada linha, os passos que o multiponto de locais armazena as localizações dos locais das espécies.

```
CREATE TABLE SPECIES_SITINGS (species  varchar(32),
                              genus varchar(32),
                              sitings  db2gse.ST_MultiPoint);

/* Criar a instrução insert do SQL para preencher as espécies, gêneros e
   sítios. Os pontos de interrogação são marcadores de parâmetros que
   indicam o
   nome e os valores da água que serão recuperados no tempo de execução. */
strcpy (shp_sql,"insert into SPECIES_SITINGS (species,genus,sitings)
values (?,?, db2gse.MPointFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Fazer o bind do valor de espécies varchar ao primeiro parâmetro. */
pcbvalue1 = species_len;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, species_len, 0, species, species_len, &pcbvalue1);

/* Fazer o bind do valor do gênero varchar ao segundo parâmetro. */
pcbvalue2 = genus_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, genus_len, 0, name, genus_len, &pcbvalue2);

/* Fazer o bind do shape ao terceiro parâmetro. */
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, sitings_len, 0, sitings_shape, sitings_len, &pcbvalue3);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

MPolyFromShape

MPolyFromShape toma um shape do tipo multipolígono e uma identidade do sistema de referência espacial para retornar um multipolígono.

Sintaxe

db2gse.MPolyFromShape(ShapeMultiPolygon Blob(1m), srs db2gse.coordref)

Tipo de retorno

db2gse.ST_MultiPolygon

Exemplos

Este fragmento de código preenche a tabela LOTS.

A tabela LOTS armazena a id do lote que identifica exclusivamente cada lote e o multipolígono do lote que contém a geometria de linha do lote.

```
CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );

/* Criar a instrução insert do SQL para preencher a lot_id e o lote. Os
   pontos de interrogação são marcadores de parâmetro que indicam a
   lot_id e os valores do
   lote que serão recuperados no tempo de execução. */
strcpy (shp_sql,"insert into LOTS (lot_id,lot)
values (?, db2gse.MPolyFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da id do lote ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Fazer o bind do shape do lote ao segundo parâmetro. */
pcbvalue2 = lot_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, lot_len, 0, lot_shape, lot_len, &pcbvalue2);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

PointFromShape

PointFromShape toma um shape do tipo ponto e uma identidade do sistema de referência espacial para retornar um ponto.

Sintaxe

db2gse.PointFromShape(db2gse.ShapePoint blob(1M), srs db2gse.coordref)

Tipo de retorno

db2gse.ST_Point

Exemplos

O fragmento do programa preenche a tabela HAZARDOUS_SITES.

Os locais arriscados estão armazenados na tabela HAZARDOUS_SITES criada com a instrução CREATE TABLE que segue. O nome da localização, definido como um ponto, armazena uma localização que é o centro geográfico de cada ponto arriscado.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);

/* Criar a instrução insert do SQL para preencher a id do local, nome e
   localização. Os pontos de interrogação são marcadores de parâmetros
   que indicam a
   id do local, nome e os valores de localização que serão recuperados no
   tempo de execução. */
strcpy (shp_sql,"insert into HAZARDOUS_SITES (site_id, name, location)
values (?,?, db2gse.PointFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da id do local ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
   SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);

/* Fazer o bind do valor varchar do nome ao segundo parâmetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Fazer o bind do shape de localização ao terceiro parâmetro. */
pcbvalue3 = location_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, location_len, 0, location_shape, location_len, &pcbvalue3);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

PolyFromShape

PolyFromShape toma um shape do tipo polígono e uma identidade do sistema de referência espacial para retornar um polígono.

Sintaxe

db2gse.PolyFromShape (ShapePolygon Blob(1M), srs db2gse.coordref)

Tipo de retorno

db2gse.ST_Polygon

Exemplos

O fragmento do programa preenche a tabela SENSITIVE_AREAS. Os pontos de interrogação representam marcadores de parâmetros para ao id, nome, tamanho, tipo e valores da zona que serão recuperados no tempo de execução.

A tabela SENSITIVE_AREAS contém várias colunas que descrevem as instituições ameaçadas além da coluna zona que armazena a geometria do polígono da instituição.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);

/* Criar a instrução insert do SQL para preencher a id, nome, tamanho, tipo e
   zona. Os pontos de interrogação são marcadores de parâmetro que
   indicam o  id, nome, tamanho, tipo e valores da zona que serão
   recuperados no tempo de execução. */
strcpy (shp_sql,"insert into SENSITIVE_AREAS (id, name, size, type, zone)
values (?,?,?,?, db2gse.PolyFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da id ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
   SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);
/* Fazer o bind do valor varchar do nome ao segundo parâmetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Fazer o bind da dimensão flutuante ao terceiro parâmetro. */
pcbvalue3 = 0;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_FLOAT,
```

```
     SQL_REAL, 0, 0, &size, 0, &pcbvalue3);

/* Fazer o bind do tipo varchar ao quarto parâmetro. */
pcbvalue4 = type_len;
rc = SQLBindParameter (hstmt, 4, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_VARCHAR, type_len, 0, type, type_len, &pcbvalue4);

/* Fazer o bind do polígono de zona ao quinto parâmetro. */
pcbvalue5 = zone_len;
rc = SQLBindParameter (hstmt, 5, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, zone_len, 0, zone_shp, zone_len, &pcbvalue5);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

ShapeToSQL

O ShapeToSQL constrói o valor de um db2gse.ST_Geometry dada sua reconhecida representação no modo binário. O valor 0 de SRID é usado automaticamente.

Sintaxe

db2gse.ST_ShapeToSQL(ShapeGeometry blob(1M))

Tipo de retorno

db2gse.ST_Geometry

Exemplos

O seguinte fragmento de código C contém funções ODBC incorporadas às funções SQL do DB2 Spatial Extender que inserem dados na tabela LOTS. A tabela LOTS foi criada com duas colunas: a lot_id, que identifica exclusivamente cada um dos lotes, e a coluna multipolígono do lote, que contém a geometria de cada lote.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

A função ShapeToSQL converte shapes em geometria do DB2 Spatial Extender. A instrução INSERT inteira é copiada em shp_sql. A instrução INSERT contém marcadores do parâmetro para aceitar os dados da id do lote e os dados do LOT dinamicamente.

```
/* Criar a instrução insert do SQL para preencher a id do lote e o
  multipolígono do lote. Os pontos de interrogação são marcadores
  do parâmetro que  indicam a id do_lote e os valores do lote
  que serão recuperados no    tempo de execução.*/

strcpy (shp_sql,"insert into lots (lot_id, lot) values(?,
db2gse.ShapeToSQL(cast(? as blob(1m))))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */

rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */

rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da chave ao primeiro parâmetro.*/

pcbvalue1 = 0;

rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Fazer o bind do shape ao segundo parâmetro. */
```

```
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Executar a instrução insert . */

rc = SQLExecute (hstmt);
```

ST_Area

ST_A área toma um polígono ou multipolígonos e retorna sua área.

Sintaxe

db2gse.ST_Area(s db2gse.ST_Surface)

Tipo de retorno

Duplo

Exemplos

O engenheiro da cidade precisa de uma lista de áreas de construção. Para obtê-la, um técnico GIS seleciona a ID da construção e a área de cada base do prédio.

As bases da construção são armazenadas na tabela BUILDINGFOOTPRINTS que foi criada com a seguinte instrução CREATE TABLE:

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

Para atender ao pedido do engenheiro da cidade, o técnico utiliza a seguinte instrução SELECT para selecionar a chave exclusiva, a id da construção e a área de cada base de construção a partir da tabela BUILDINGFOOTPRINTS:

```
SELECT building_id, db2gse.ST_Area (footprint) "Area"
FROM BUILDINGFOOTPRINTS;
```

A instrução SELECT retorna o seguinte conjunto de resultados:

```
ID_da construção  Área
------------  ------------------------
         506    +1.40768000000000E+003
        1208    +2.55759000000000E+003
         543    +1.80786000000000E+003
         178    +2.08671000000000E+003
         .
         .
         .
```

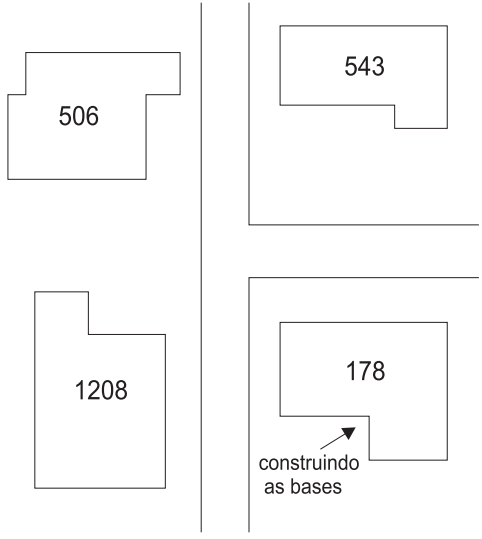


Figura 28. Utilizando a área para encontrar a base da construção. Quatro das bases de construção rotuladas com seus números de ID de construção são exibidas junto à rua adjacente.

ST_AsBinary

ST_AsBinary toma um objeto de geometria e retorna sua representação de modo binário reconhecida.

Sintaxe

db2gse.ST_AsBinary(g db2gse.ST_Geometry)

Tipo de retorno

BLOB(1m)

Exemplos

O seguinte fragmento de código ilustra como a função ST_AsBinary converte os multipolígonos da base da tabela BUILDINGFOOTPRINTS em multipolígonos WKB. Tais multipolígonos são passados para a função de desenho do polígono (draw_polygon) para exibição.

```
/* Criar a expressão SQL. */
strcpy(sqlstmt, "select db2gse.ST_AsBinary (footprint) from BUILDINGFOOTPRINTS
where db2gse.EnvelopesIntersect(footprint, db2gse.ST_PolyFromWKB
(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Preparar as instruções SQL. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* Definir o comprimento de pcbvalue1 do shape. */
pcbvalue1 = blob_len;

/* Fazer o bind do parâmetro do shape */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB, blob_len,
0, shape, blob_len, &pcbvalue1);

/* Executar a consulta */
rc = SQLExecute (hstmt);

/* Atribuir os resultados da consulta (os polígonos da Zone) a
  fetched_binary variable.
 */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* Buscar cada polígono dentro da janela de exibição e exibi-lo. */
while(SQL_SUCCESS == (rc = SQLFetch(hstmt)))
  draw_polygon(fetched_binary);
```

ST_AsText

O db2gse.ST_AsText pega um objeto de geometria e retorna sua representação de texto reconhecida.

Sintaxe

db2gse.ST_AsText(g db2gse.ST_Geometry)

Tipo de retorno

Varchar(4000)

Exemplos

No cenário a seguir, a função db2gse.ST_AsText converte o ponto da localização HAZARDOUS_SITES na descrição de seu texto:

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(40),
                              location  db2gse.ST_Point);
INSERT INTO HAZARDOUS_SITES
VALUES (102,
        'W. H. Kleenare Chemical Repository',
        db2gse.ST_PointFromText('point (1020.12 324.02)',
db2gse.coordref()..srid(0)));

SELECT site_id, name, cast(db2gse.ST_AsText(location) as varchar(40))
    "Location"
FROM HAZARDOUS_SITES;
```

A instrução SELECT retorna o seguinte conjunto de resultados:

```
ID do_LOCAL Nome                              Localização
------- ------------------------------------ ---------------------------------
    102 W. H. Kleenare Chemical Repository   POINT (1020.00000000 324.00000000)
```

ST_Boundary

ST_Boundary toma um objeto de geometria e retorna seu limite combinado como um objeto de geometria.

Sintaxe

db2gse.ST_Boundary(g db2gse.ST_Geometry)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

No seguinte fragmento de código, é criada uma tabela chamada BOUNDARY_TEST. BOUNDARY_TEST tem duas colunas: GEOTYPE, que está definida como varchar e G1, definida como geometria de superclasse. As instruções INSERT que se seguem inserem cada uma das geometrias de subclasse. A função ST_Boundary recupera o limite de cada subclasse que está armazenada na coluna de geometria G1. Observe que a dimensão da geometria resultante é sempre uma menor que a geometria de entrada. Os pontos e multipontos sempre resultam num limite que é uma geometria vazia, dimensão 1. As cadeias de linhas e cadeias de linhas múltiplas retornam um limite multiponto, dimensão 0. Um polígono ou multipolígono sempre retorna um limite de cadeia de linhas múltiplas, dimensão 1.

```
CREATE TABLE BOUNDARY_TEST (GEOTYPE varchar(20), G1 db2gse.ST_Geometry)

INSERT INTO BOUNDARY_TEST
VALUES('Point',
       db2gse.ST_PointFromText('point (10.02 20.01)',
                     db2gse.coordref()..srid(0)))

INSERT INTO BOUNDARY_TEST
VALUES ('Linestring',
       db2gse.ST_LineFromText('linestring (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO BOUNDARY_TEST
VALUES('Polygon',
       db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,25.02 34.15,
                                  19.15 33.94, 10.02 20.01))',
                    db2gse.coordref()..srid(0)))

INSERT INTO BOUNDARY_TEST
VALUES('Multipoint',
       db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO BOUNDARY_TEST
VALUES('Multilinestring',
       db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
11.92 25.64), (9.55 23.75,15.36 30.11))',
                     db2gse.coordref()..srid(0)))
```

```
INSERT INTO BOUNDARY_TEST
VALUES('Multipolygon',
       db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
                                    25.02 34.15, 19.15 33.94,10.02 20.01)),
                                        ((51.71 21.73,73.36 27.04,71.52 32.87,
                                          52.43 31.90,51.71 21.73)))',
                     db2gse.coordref()..srid(0)))

SELECT GEOTYPE,
       CAST(db2gse.ST_AsText(db2gse.ST_Boundary (G1)) as varchar(280))
      "The boundary"
FROM BOUNDARY_TEST


GEOTYPE              The boundary
-------------------- ------------------------------------------------------------
Point                POINT EMPTY
Linestring           MULTIPOINT ( 10.02000000 20.01000000, 11.92000000
 25.64000000)
Polygon              MULTILINESTRING (( 10.02000000 20.01000000, 19.15000000
 33.94000000, 25.02000000 34.15000000, 11.92000000 35.64000000, 10.02000000
 20.01000000))
Multipoint           POINT EMPTY
Multilinestring      MULTIPOINT ( 9.55000000 23.75000000, 10.02000000
 20.01000000, 11.92000000 25.64000000, 15.36000000 30.11000000)
Multipolygon         MULTILINESTRING (( 51.71000000 21.73000000, 73.36000000
 27.04000000, 71.52000000 32.87000000, 52.43000000 31.90000000, 51.71000000
 21.73000000),( 10.02000000 20.01000000, 19.15000000 33.94000000, 25.02000000
 34.15000000, 11.92000000 35.64000000, 10.02000000 20.01000000))

  6 record(s) selected.
```

ST_Buffer

ST_Buffer toma um objeto de geometria e distância e retorna o objeto de geometria que está ao redor do objeto de origem.

Sintaxe

db2gse.ST_Buffer(g db2gse.ST_Geometry , adistance Double)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

O Supervisor do Município precisa de uma lista de locais arriscados num raio de 8 km que abranja áreas sensíveis, como escolas, hospitais e casas de repousos. As áreas sensíveis são armazenadas na tabela SENSITIVE_AREAS que é criada com a seguinte instrução CREATE TABLE. A coluna ZONE é definida como um polígono, que está armazenado como o contorno de cada linha sensível.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

As áreas de risco são armazenadas na tabela HAZARDOUS_SITES que foi criada com a seguinte instrução CREATE TABLE. A coluna LOCATION, definida como um ponto, armazena uma localização que é o centro geográfico de cada local arriscado.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

As tabelas SENSITIVE_AREAS e HAZARDOUS_SITES são unidas pela função db2gse.ST_Overlaps. A função retorna 1 (VERDADEIRO) para todas as linhas SENSITIVE_AREAS cujos polígonos da zona sobrepõem o raio de buffer de 8 km do ponto de localização do HAZARDOUS_SITES.

```
SELECT sa.name "Sensitive Areas", hs.name "Hazardous Sites"
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Overlaps(sa.zone, db2gse.ST_Buffer (hs.location,(5 * 5280)))
=1;
```

Na Figura 29 na página 195, algumas áreas sensíveis neste distrito administrativo estão dentro do buffer de 8 km das localizações de pontos arriscados. Os dois buffers de 8 km fazem interseção com o hospital e um deles faz interseção com a escola. Contudo, a casa de repouso está localizada com segurança fora dos dois raios.

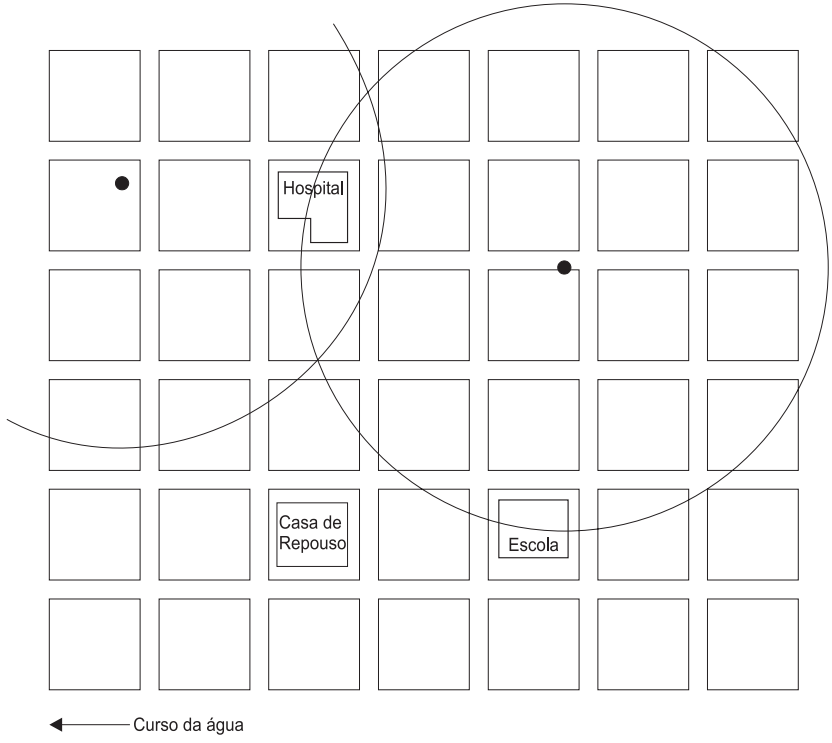


Figura 29. Um buffer com um raio de 8 km é aplicado a um ponto

ST_Centroid

ST_Centroid toma um polígono ou um multipolígono e retorna seu centro geométrico como um ponto.

Sintaxe

db2gse.ST_Centroid(s db2gse.ST_Surface)
db2gse.ST_Centroid(ms db2gse.ST_MultiSurface)

Tipo de retorno

Na superfície: db2gse.ST_Point

Exemplos

O técnico em GIS da cidade deseja exibir os multipolígonos das bases da construção como pontos únicos num gráfico de densidade de construção.

As bases da construção são armazenadas na tabela BUILDINGFOOTPRINTS que foi criada com a seguinte instrução CREATE TABLE.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                        footprint   db2gse.ST_MultiPolygon);
```

A função ST_Centroid retorna a id de centro de cada multipolígono da base da construção. A função AsBinaryShape converte o ponto da id de centro em um shape, a representação externa que é reconhecida pela aplicação.

```
SELECT building_id,
       CAST(db2gse.AsBinaryShape(db2gse.ST_Centroid (footprint)) as blob(1m))
"Centroid"
FROM BUILDINGFOOTPRINTS;
```

ST_Contains

ST_Contains toma dois objetos de geometria e retorna 1 (VERDADEIRO) se o primeiro objeto incluir completamente o segundo; caso contrário, retornará 0 (FALSO).

Sintaxe

db2gse.ST_Contains(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

No exemplo abaixo são criadas duas tabelas. Uma tabela contém as bases de construção da cidade, ao passo que a outra contém seus lotes. O engenheiro da cidade deseja verificar se todas as bases de construção estão inteiramente dentro de seus lotes.

Nas duas tabelas o tipo de dados de multipolígono armazena a geometria das bases e lotes de construção. O projetista de bancos de dados selecionou multipolígonos para ambos os recursos. O projetista imaginou que os lotes podem ser desmembrados por recursos naturais, como um rio e que as bases de construção podem ser geralmente constituídas de vários edifícios.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                        footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

O engenheiro da cidade primeiro seleciona as construções que não estão inteiramente dentro de um lote.

```
SELECT building_id
  FROM BUILDINGFOOTPRINTS, LOTS
WHERE db2gse.ST_Contains(lot,footprint) = 0;
```

O engenheiro da cidade imagina que a primeira consulta retornará uma lista de todas as IDs de construção que possuem bases fora de um polígono do lote. Mas o engenheiro da cidade também sabe que estas informações não indicarão se as outras construções possuem a ID de lote correta atribuída a eles. Esta segunda consulta executa uma verificação de integridade dos dados na id do lote da tabela BUILDINGFOOTPRINTS.

```
SELECT bf.building_id "building id", bf.lot_id "buildings lot_id",
       LOTS.lot_id "LOTS lot_id"
FROM BUILDINGFOOTPRINTS bf, LOTS
WHERE db2gse.ST_Contains(lot,footprint) = 1 AND LOTS.lot_id <> bf.lot_id;
```

Na Figura 30 na página 198, as bases de construção rotuladas com suas IDs de construção estão dentro de seus lotes. As linhas do lote estão ilustradas com

linhas pontilhadas. Embora não ilustradas, estas linhas ampliam a linha central da rua e incluem totalmente os lotes e as bases de construção dentro delas.

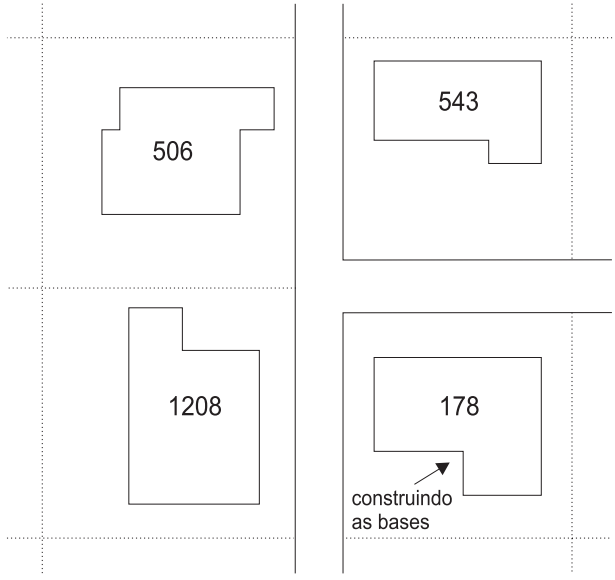


Figura 30. Utilizando ST_Contains para garantir que todas as construções fiquem dentro de seus lotes

ST_ConvexHull

ST_ConvexHull toma um objeto de geometria e retorna a carcaça convexa.

Sintaxe

db2gse.ST_ConvexHull(g db2gse.ST_Geometry)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

O exemplo cria a tabela CONVEXHULL_TEST que tem duas colunas: GEOTYPE e G1. A coluna GEOTYPE, varchar(20), armazenará o nome da subclasse de geometria que está armazenada em G1, a qual está definida como geometria.

```
CREATE TABLE CONVEXHULL_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Cada instrução INSERT insere uma geometria de cada tipo de subclasse na tabela CONVEXHULL_TEST.

```
INSERT INTO CONVEXHULL_TEST
VALUES('Point',
       db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES ('Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES('Polygon',
       db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,25.02 34.15,
                                         19.15 33.94,10.02 20.01))',
                    db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES('Multipoint',
       db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES('Multilinestring',
       db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
       11.92 25.64),(9.55 23.75,15.36 30.11))',
                     db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES('Multipolygon',
       db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
       25.02 34.15, 19.15 33.94,10.02 20.01)),
```

```
                                        ((51.71 21.73,73.36 27.04,71.52 32.87,
                                          52.43 31.90,51.71 21.73)))',
                        db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT relaciona o nome da subclasse armazenado na coluna GEOTYPE e a carcaça convexa. A carcaça convexa gerada pela função ST_ConvexHull é convertida em texto pela função ST_AsText. É então feita a conversão em um varchar(256) porque a saída padrão de ST_AsText é varchar(4000).

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText(db2gse.ST_ConvexHull(G1)))
as varchar(256) "The convexhull"
FROM CONVEXHULL_TEST
```

ST_CoordDim

ST_CoordDim retorna as dimensões coordenadas do valor ST_Geometry. Para obter uma explicação sobre as dimensões coordenadas, consulte o “Pontos” na página 133.

Sintaxe

db2gse.ST_CoordDim(g1 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A tabela coorddim_test é criada com as colunas geotype e g1. A coluna geotype armazena o nome da subclasse de geometria armazenada na coluna de geometria g1.

```
CREATE TABLE coorddim_test (geotype varchar(20), g1 db2gse.ST_Geometry)
```

As instruções INSERT inserem uma subclasse de amostra na tabela coorddim_test.

```
INSERT INTO coorddim_test VALUES(
    'Point', db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
       11.92 25.64)',
db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Polygon', db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,
25.02 34.15, 19.15 33.94,10.02 20.01))', db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Multipoint', db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Multilinestring', db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,
10.32 23.98,11.92 25.64),(9.55 23.75,15.36 30.11))',
db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Multipolygon',
```

```
MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,25.02 34.15,
19.15 33.94,10.02 20.01)),((51.71 21.73,73.36 27.04,71.52 32.87,
52.43 31.90,51.71 21.73)))', db2gse.coordref()..srid(0))
)
```

A instrução SELECT relaciona o nome da subclasse armazenada na coluna geotype à dimensão coordenada desse geotype.

```
SELECT geotype, db2gse.ST_coordDim(g1)' coordinate_dimension'
FROM coorddim_test

GEOTYPE              coordinate_dimension
-------------------- -------------------------
ST_Point                                   2
ST_Linestring                              2
ST_Polygon                                 2
ST_Multipoint                              2
ST_Multilinestring                         2
ST_Multipolygon                            2

  6 record(s) selected.
```

ST_Crosses

ST_Crosses toma dois objetos de geometria e retorna 1 (VERDADEIRO) se sua interseção resultar num objeto de geometria cuja dimensão seja menor que a dimensão máxima dos objetos de origem. O objeto de interseção contém pontos que são interiores para as duas geometrias de origem e não é igual a nenhum dos objetos de origem. Do contrário, retornará 0 (FALSO).

Sintaxe

db2gse.ST_Crosses(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A prefeitura do município está considerando uma nova lei que declara que todas as estações de armazenamento de lixo tóxico dentro do município não poderão estar dentro de 8 km de qualquer curso de água. O gerente GIS do município possui uma representação exata de rios e córregos, que está armazenada como cadeias de linhas múltiplas na tabela WATERWAYS. Mas, o gerente GIS apresenta somente uma única localização de ponto para cada estação de armazenamento de lixo tóxico.

```
CREATE TABLE WATERWAYS (id     integer,
                        name   varchar(128),
                        water  db2gse.ST_MultiLineString);

CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                               name      varchar(128),
                               location  db2gse.ST_Point);
```

Para determinar se o supervisor do município precisa ser alertado das estações que violam o regulamento proposto, o gerente GIS deve ter buffer dos locais arriscados e verificar se algum rio ou córrego cruza o polígono do buffer. O predicado ST_Crosses compara os HAZARDOUS_SITES com buffer aos WATERWAYS. Assim, retornam somente esses registros nos quais o curso da água cruza o raio regulamentado proposto pelo município.

```
SELECT ww.name "River or stream", hs.name "Hazardous site"
FROM WATERWAYS ww, HAZARDOUS_SITES hs
WHERE db2gse.ST_Crosses(db2gse.ST_Buffer(hs.location,(5 * 5280)),ww.water)
=1;
```

Na Figura 31 na página 204, o buffer de 8 km dos locais de lixo tóxico cruza a rede de córregos que passa pelo distrito administrativo do município. A rede de córregos foi definida como uma cadeia de linhas múltiplas. Portanto, o conjunto de resultados inclui segmentos da cadeia de linha que fazem parte destes segmentos que cruzam o raio.

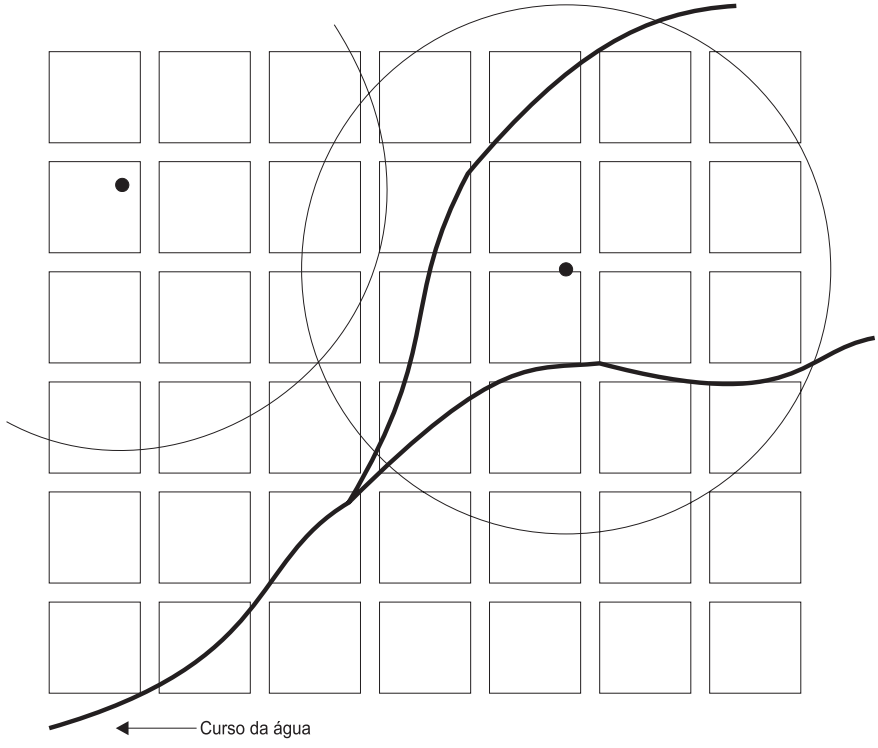


Figura 31. Utilizando ST_Crosses para encontrar cursos de água que passam por área de lixo tóxico

ST_Difference

ST_Difference toma dois objetos de geometria e retorna um objeto de geometria que é a diferença dos objetos de origem.

Sintaxe

db2gse.ST_Difference(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

O engenheiro da cidade precisa saber a área total da área de lotes da cidade não abrangida por uma construção. Ou seja, o engenheiro da cidade deseja a soma da área de lotes após a remoção da área de construção.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                         footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS ( lot_id   integer,
                        lot   db2gse.ST_MultiPolygon);
```

O engenheiro da cidade reune a tabela BUILDINGFOOTPRINTS e LOTS no lot_id. O engenheiro então soma a área de diferença dos lotes menos as bases de construção.

```
SELECT SUM(db2gse.ST_Area(db2gse.ST_Difference(lot,footprint)))
FROM BUILDINGFOOTPRINTS bf, LOTS
WHERE bf.lot_id = LOTS.lot_id;
```

ST_Dimension

ST_Dimension toma um objeto de geometria e retorna sua dimensão.

Sintaxe

db2gse.ST_Dimension(g1 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A tabela DIMENSION_TEST é criada com as colunas GEOTYPE e G1. A coluna GEOTYIPE armazena o nome da subclasse de geometria armazenada na coluna de geometria G1.

```
CREATE TABLE DIMENSION_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

As instruções INSERT inserem uma subclasse de amostra na tabela DIMENSION_TEST.

```
INSERT INTO DIMENSION_TEST
VALUES('Point',
       db2gse.ST_PointFromText('point (10.02 20.01)',

db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES ('Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES('Polygon',
       db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,25.02 34.15,
                                    19.15 33.94,10.02 20.01))',
                  db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES('Multipoint',
       db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES('Multilinestring',
       db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
              11.92 25.64),(9.55 23.75,15.36 30.11))',
                     db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES('Multipolygon',
       db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
```

```
             25.02 34.15,19.15 33.94,10.02 20.01)),
                                    ((51.71 21.73,73.36 27.04,71.52 32.87,
                                      52.43 31.90,51.71 21.73)))',
db2gse.coordref)..srid(0)))
```

A seguinte instrução SELECT relaciona o nome da subclasse armazenado na coluna GEOTYPE à dimensão desse geotipo.

```
SELECT geotype, db2gse.ST_Dimension(g1) "A dimensão"
FROM DIMENSION_TEST
```

O seguinte conjunto de resultados retorna.

```
GEOTYPE              The dimension
-------------------- -------------
ST_Point                       0
ST_Linestring                  1
ST_Polygon                     2
ST_Multipoint                  0
ST_Multilinestring             1
ST_Multipolygon                2

  6 record(s) selected.
```

ST_Disjoint

ST_Disjoint toma duas geometrias e retorna 1 (VERDADEIRO) se a interseção de duas geometrias produzir um conjunto vazio; do contrário retornará 0 (FALSO).

Sintaxe

db2gse.ST_Disjoint(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

Uma companhia de seguros precisa orçar a cobertura de seguro para os hospitais, casas de repouso e escolas de uma cidade. Parte deste processo inclui a determinação da ameaça que os locais de lixo tóxico representam para cada instituição. A companhia de seguro pretende considerar somente as instituições que estejam com risco de contaminação. O consultor GIS contratado pela companhia de seguros foi encarregado de localizar todas as instituições que não estejam dentro do raio de 8 km de um depósito de lixo tóxico.

A tabela SENSITIVE_AREAS contém várias colunas que descrevem as instituições ameaçadas além da coluna ZONE, que armazena a geometria do polígono da instituição.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

A tabela HAZARDOUS_SITES armazena a identidade dos locais nas colunas SITE_ID e NAME, ao passo que a localização geográfica real de cada local está armazenada na coluna LOCATION.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

A seguinte instrução SELECT relaciona os nomes de todas as áreas sensíveis que não estejam dentro do raio de 8 km de um local de lixo tóxico. A função ST_Intersects pode substituir a função ST_Disjoint nesta consulta se o resultado da função estiver definido igual a 0 em vez de 1. Isto porque ST_Intersects e ST_Disjoint retornam o resultado oposto exato.

```
SELECT sa.name
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Disjoint(db2gse.ST_Buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

Na Figura 32, são sensíveis os locais comparados ao raio de 8 km do lixo tóxico. A casa de repouso é a única área sensível em que a função ST_Disjoint retornará 1 (VERDADEIRO). A função ST_Disjoint retorna 1 sempre que duas geometrias não fazem interseção de alguma forma.

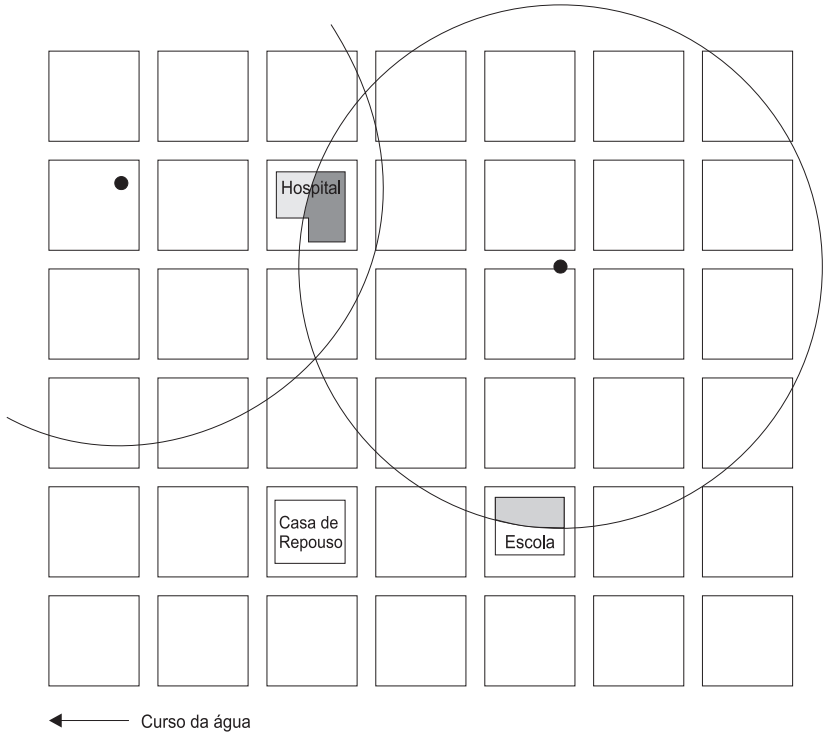


Figura 32. Usando ST_Disjoint para encontrar as construções que não estejam dentro de (interseção) alguma área de lixo tóxico

ST_Distance

ST_Distance toma duas geometrias e retorna a distância mais próxima que as separa.

Sintaxe

db2gse.ST_Distance(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Duplo

Exemplos

O engenheiro da cidade precisa de uma lista de todas as construções que estejam dentro de 30 cm de qualquer linha do lote.

A coluna BUILDINGI_ID da tabela BUILDINGFOOTPRINTS identifica exclusivamente cada construção. A coluna LOT_ID identifica o lote a que cada construção pertence. O multipolígono da base armazena a geometria de cada base da construção.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                        footprint   db2gse.ST_MultiPolygon);
```

A tabela LOTS armazena a ID do lote que identifica exclusivamente cada lote e o multipolígono do lote que contém a geometria de linha do lote.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot      db2gse.ST_MultiPolygon);
```

A consulta retorna uma lista de IDs de construção que estejam dentro de 30 cm das linhas do lote. A função ST_Distance executa uma junção espacial entre as bases e o limite dos multipolígonos do lote. No entanto, a equijunção entre bf.lot_id LOTS.lot_id assegura que somente os multipolígonos pertencentes ao mesmo lote sejam comparados pela função ST_Distance.

```
SELECT bf.building_id
FROM BUILDINGFOOTPRINTS bf, LOTS
 WHERE bf.lot_id = LOTS.lot_id AND
       db2gse.ST_Distance(bf.footprint, db2gse.ST_Boundary(LOTS.lot)) <= 1.0;
```

ST_Endpoint

ST_Endpoint toma uma cadeia de linha e retorna um ponto que é o último da cadeia.

Sintaxe

db2gse.ST_Endpoint(c db2gse.ST_Curve)

Tipo de retorno

db2gse.ST_Point

Exemplos

A tabela ENDPOINT_TEST armazena a coluna inteira de GID que identifica exclusivamente cada linha na cadeia de linha LN1 que armazena cadeias de linhas.

```
CREATE TABLE ENDPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

As instruções INSERT inserem cadeias de linhas na tabela ENDPOINT_TEST. A primeira não possui coordenadas Z ou medidas; a segunda possui.

```
INSERT INTO ENDPOINT_TEST
VALUES( 1,
        db2gse.ST_LineFromText('linestring (10.02 20.01,23.73 21.92,
        30.10 40.23)', db2gse.coordref()..srid(0)))

INSERT INTO ENDPOINT_TEST
VALUES (2,
       db2gse.ST_LineFromText('linestring zm (10.02 20.01 5.0 7.0,
       23.73 21.92 6.5 7.1, 30.10 40.23 6.9 7.2)',
                    db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT relaciona a coluna GID à saída da função ST_Endpoint. A função ST_Endpoint gera uma geometria de ponto que é convertida em texto pela função ST_AsText. A função CAST é usada para abreviar o valor varchar(4000) da função ST_AsText para um varchar(60).

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.ST_Endpoint(ln1)) AS varchar(60))
"Endpoint"
FROM ENDPOINT_TEST
```

O seguinte conjunto de resultados retorna.

```
GID         Endpoint
----------- ------------------------------------------------------------
          1 POINT ( 30.10000000 40.23000000)
          2 POINT ZM ( 30.10000000 40.23000000 7.00000000 7.20000000)

  2 record(s) selected.
```

ST_Envelope

ST_Envelope toma um objeto de geometria e retorna seu quadro delimitador como uma geometria.

Sintaxe

db2gse.ST_Envelope(g db2gse.ST_Geometry)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

A coluna GEOTYPE na tabela ENVELOPE_TEST armazena o nome da subclasse de geometria armazenada na coluna de geometria G1.

```
CREATE TABLE ENVELOPE_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

As seguintes instruções INSERT inserem cada subclasse de geometria na tabela ENVELOPE_TEST.

```
INSERT INTO ENVELOPE_TEST
VALUES('Point',
       db2gse.ST_PointFromText('point (10.02 20.01)',

db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES ('Linestring',
        db2gse.ST_LineFromText('linestring (10.01 20.01, 10.01 30.01,
        10.01 40.01)', db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES ('Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
       11.92 25.64)',
       db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Polygon',
       db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,25.02 34.15,
                                         19.15 33.94,10.02 20.01))',
                    db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Multipoint',
       db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
       11.92 25.64)',
       db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Multilinestring',
       db2gse.ST_MLineFromText('multilinestring ((10.01 20.01,20.01 20.01,
       30.01 20.01), (30.01 20.01,40.01 20.01,50.01 20.01))',
                    db2gse.coordref()..srid(0)))
```

```
INSERT INTO ENVELOPE_TEST
VALUES('Multilinestring',
       db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
       11.92 25.64), ( 9.55 23.75,15.36 30.11))',
                     db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Multipolygon',
       db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
       25.02 34.15, 19.15 33.94,10.02 20.01)),
                                    ((51.71 21.73,73.36 27.04,71.52 32.87,
                                      52.43 31.90,51.71 21.73)))',
                     db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT relaciona o nome da subclasse ao lado de seu envelope. Como a função ST_Envelope retorna um ponto, uma cadeia de linha ou um polígono, sua saída será convertida em texto com a função ST_AsText. A função CAST converte o resultado padrão varchar(4000) da função ST_AsText para um varchar(280).

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText(db2gse.ST_Envelope(g1)) AS varchar(280))
"The envelope"
FROM ENVELOPE_TEST
```

O seguinte conjunto de resultados retorna.

```
GEOTYPE              The envelope
-------------------- ------------------------------------------------------------
Point                POINT ( 10.02000000 20.01000000)
Linestring           LINESTRING ( 10.01000000 20.01000000, 10.01000000
 40.01000000)
Linestring           POLYGON (( 10.02000000 20.01000000, 11.92000000
 20.01000000, 11.92000000 25.64000000, 10.02000000 25.64000000, 10.02000000
 20.01000000))
Polygon              POLYGON (( 10.02000000 20.01000000, 25.02000000
 20.01000000, 25.02000000 35.64000000, 10.02000000 35.64000000, 10.02000000
 20.01000000))
Multipoint           POLYGON (( 10.02000000 20.01000000, 11.92000000
 20.01000000, 11.92000000 25.64000000, 10.02000000 25.64000000, 10.02000000
 20.01000000))
Multilinestring      LINESTRING ( 10.01000000 20.01000000, 50.01000000
 20.01000000)
Multilinestring      POLYGON (( 9.55000000 20.01000000, 15.36000000
 20.01000000, 15.36000000 30.11000000, 9.55000000 30.11000000, 9.55000000
 20.01000000))
Multipolygon         POLYGON (( 10.02000000 20.01000000, 73.36000000
 20.01000000, 73.36000000 35.64000000, 10.02000000 35.64000000, 10.02000000
 20.01000000))

  8 record(s) selected.
```

ST_Equals

ST_Equals compara duas geometrias e retorna 1 (VERDADEIRO) se as geometrias são idênticas; do contrário, retorna 0 (FALSO).

Sintaxe

db2gse.ST_Equals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

O técnico GIS da cidade suspeita que alguns dados na tabela BUILDINGFOOTPRINTS foram duplicados de alguma forma. O técnico consulta a tabela para determinar se algum multipolígono da base é igual.

A tabela BUILDINGFOOTPRINTS foi criada com a seguinte instrução. A coluna BUILDING_ID identifica as construções; a coluna LOT_ID identifica o lote da construção e a coluna FOOTPRINT armazena a geometria da construção.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                        footprint   db2gse.ST_MultiPolygon);
```

A tabela BUILDINGFOOTPRINTS é unida a si própria espacialmente pelo predicado ST_Equals, que retorna 1 sempre que encontra dois multipolígonos que são iguais. A condição bf1.building_id <> bf2.building_id é necessária para eliminar a comparação da mesma geometria.

```
SELECT bf1.building_id, bf2.building_id
FROM BUILDINGFOOTPRINTS bf1, BUILDINGFOOTPRINTS bf2
WHERE db2gse.ST_Equals(bf1.footprint,bf2.footprint) = 1
      e bf1.building_id <> bf2.building_id;
```

ST_ExteriorRing

ST_ExteriorRing toma um polígono e retorna seu anel externo como uma cadeia de linha.

Sintaxe

db2gse.ST_ExteriorRing(s db2gse.ST_Polygon)

Tipo de retorno

db2gse.ST_LineString

Exemplos

Um ornitologista que está estudando a população de pássaros em várias ilhas marinhas do sul, sabe que a zona de alimentação de uma determinada espécie de pássaro restringe-se ao litoral. Para calcular a capacidade de transporte da ilha, o ornitologista solicita o perímetro da ilha. Embora algumas ilhas tenham vários lagos, as linhas de contorno dos lagos são habitadas exclusivamente por outras espécies de pássaros mais agressivas. Portanto, ele solicita o perímetro do anel externo das ilhas.

As colunas ID e NAME da tabela ISLANDS identificam cada ilha e a coluna LAND do tipo ST_Polygon armazena a geometria de cada uma.

```
CREATE TABLE ISLANDS (id    integer,
                      name  varchar(32),
                      land  db2gse.ST_Polygon);
```

A função ST_ExteriorRing extrai o anel externo de cada polígono da ilha como uma cadeia de linha. O comprimento da cadeia de linha é estabelecido pela função comprimento. Os comprimentos das cadeias de linha são resumidos pela função SUM.

```
SELECT SUM(db2gse.ST_length(db2gse.ST_ExteriorRing (land))) FROM ISLANDS;
```

Na Figura 33 na página 216, os anéis externos das ilhas representam a interface ecológica que cada ilha compartilha com o mar. Algumas ilhas têm lagos, que são representados pelos anéis interiores dos polígonos.

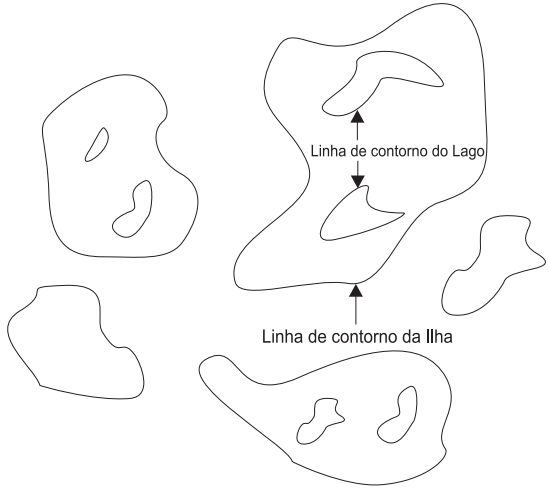


Figura 33. Utilizando ST_ExteriorRing para determinar o comprimento do contorno da ilha

ST_GeometryFromText

O ST_GeometryFromText pega a representação de um texto conhecido e a identidade de um sistema de referência espacial e apresenta um objeto da geometria.

Sintaxe

db2gse.ST_GeometryFromText(geometryTaggedText Varchar(4000), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

A tabela GEOMETRY_TEST contém a coluna GID, que identifica exclusivamente cada linha e a coluna G1, que armazena a geometria.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

As instruções INSERT inserem os dados nas colunas GID e G1 da tabela GEOMETRY_TEST. A função ST_GeometryFromText converte a representação do texto de cada geometria na subclasse de instâncias do DB2 Spatial Extender correspondente.

```
INSERT INTO GEOMETRY_TEST
VALUES(1, db2gse.ST_GeometryFromText('point (10.02 20.01)',

db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
VALUES (2,
        db2gse.ST_GeometryFromText('linestring (10.01 20.01, 10.01 30.01,
        10.01 40.01)', db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
VALUES(3,
       db2gse.ST_GeometryFromText('polygon ((10.02 20.01,11.92 35.64,
       25.02 34.15,19.15 33.94,10.02 20.01))',
       db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
VALUES(4,
       db2gse.ST_GeometryFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
VALUES(5,
       db2gse.ST_GeometryFromText('multilinestring ((10.02 20.01,10.32 23.98,
                                          11.92 25.64),
                                         ( 9.55 23.75,15.36 30.11))',
                          db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
VALUES(6,
```

```
 db2gse.ST_GeometryFromText('multipolygon (((10.02 20.01,11.92 35.64,
25.02 34.15, 19.15 33.94,10.02 20.01)),
                                    ((51.71 21.73,73.36 27.04,71.52 32.87,
                                      52.43 31.90,51.71 21.73)))',
                    db2gse.coordref()..srid(0)))
```

ST_GeomFromWKB

ST_GeomFromWKB toma uma representação de modo binário reconhecida e uma identidade do sistema de referência espacial e retorna um objeto de geometria.

Sintaxe

db2gse.ST_GeomFromWKB(WKBGeometry Blob(1M), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

O seguinte fragmento de código C contém funções ODBC incorporadas às funções SQL do DB2 Spatial Extender que inserem dados na tabela LOTS.

A tabela LOTS foi criada com duas colunas: a coluna LOT_ID, que identifica cada um dos lotes exclusivamente, e a coluna multipolígono do LOT, que contém a geometria de cada lote.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot      db2gse.ST_MultiPolygon);
```

A função ST_GeomFromWKB converte representações de WKB em geometria do DB2 Spatial Extender. A instrução INSERT inteira será copiada na cadeia wkb_sql char. A instrução INSERT contém marcadores do parâmetro para a ceitar os dados do LOT_ID e os dados do LOT dinamicamente.

```
/* Criar a instrução insert do SQL para preencher a id do lote e o
  multipolígono do lote. Os pontos de interrogação são marcadores
  do parâmetro que
 indicam a id do_lote e os valores do lote
  que serão recuperados no  tempo de execução.*/
strcpy (wkb_sql,"insert into LOTS (lot_id, lot) values (?,
  db2gse.ST_GeomFromWKB

(cast(? as blob(1m)), db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da chave ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Fazer o bind do shape ao segundo parâmetro. */
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

ST_GeometryN

ST_GeometryN toma uma coleção e um índice inteiro e retorna o objeto de geometria *n* na coleção.

Sintaxe

db2gse.ST GeometryN(g db2gse.ST_GeomCollection, n Integer)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

O engenheiro da cidade precisa saber se as bases da construção estão todas dentro do primeiro polígono do multipolígono do lote.

A coluna BUILDING_ID identifica exclusivamente cada linha da tabela BUILDINGFOOTPRINTS. A coluna LOT_ID identifica o lote da construção. A coluna BASE armazena a geometria da construção.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                         footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS ( lot_id   integer,
                    lot   db2gse.ST_MultiPolygon);
```

A consulta lista BUILDINGFOOTPRINTS building_id e lot_id para todas as bases da construção que estejam dentro do primeiro polígono do lote. A função ST_GeometryN retorna o primeiro polígono de um lote na matriz de polígonos.

```
SELECT bf.building_id,bf.lot_id
FROM BUILDINGFOOTPRINTS bf,LOTS
WHERE db2gse.ST_Within(footprint, db2gse.ST_GeometryN (lot,1)) = 1
      AND bf.lot_id = LOTS.lot_id;
```

ST_GeometryType

O ST_GeometryType pega um objeto ST_Geometry e apresenta o tipo de sua geometria como a cadeia.

Sintaxe

db2gse.ST_GeometryType (g db2gse.ST_Geometry)

Tipo de retorno

Varchar(4000)

Exemplos

A tabela GEOMETRYTYPE_TEST contém a coluna de geometria G1.

```
CREATE TABLE GEOMETRYTYPE_TEST(g1 db2gse.ST_Geometry)
```

As seguintes instruções INSERT inserem cada subclasse de geometria na coluna G1.

```
INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_GeometryFromText('point (10.02 20.01)',
       db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRYTYPE_TEST
VALUES (db2gse.ST_GeometryFromText('linestring (10.01 20.01, 10.01 30.01,
        10.01 40.01)', db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_Geometrytype_test values(db2gse.ST_GeomFromText('polygon
      ((10.02 20.01,11.92 35.64,25.02 34.15,19.15 33.94, 10.02 20.01))',
                        db2gse.coordref()..srid(0))))

INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_GeometryFromText('multipoint (10.02
 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_GeometryFromText('multilinestring ((10.02 20.01,10.32 23.98,
                                          11.92 25.64),
                                         (9.55 23.75,15.36 30.11))',
                        db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_GeometryFromText('multipolygon (((10.02 20.01,11.92 35.64,
        25.02 34.15, 19.15 33.94,10.02 20.01)),
                                      ((51.71 21.73,73.36 27.04,71.52 32.87,
                                        52.43 31.90,51.71 21.73)))',
                        db2gse.coordref()..srid(0)))
```

A seguinte instrução INSERT insere o tipo de geometria de cada subclasse que está armazenada na coluna de geometria G1.

```
SELECT db2gse.ST_GeometryType(g1) "Geometry type" FROM GEOMETRYTYPE_TEST
```

O seguinte conjunto de resultados retorna.

```
Geometry type
------------------------------
ST_Point
ST_LineStringST_Polygon
ST_MultiPointST_MultiLineStringST_MultiPolygon
  6 record(s) selected.
```

ST_InteriorRingN

Retorna o *n* anel interior de um polígono como uma cadeia de linha. Os anéis não estão organizados por orientação geométrica. Estão organizados de acordo com as regras definidas pelas rotinas de verificação de geometria interna. Assim, não é possível predefinir a ordem dos anéis.

Sintaxe

ST_InteriorRingN(p ST_Polygon, n Integer)

Tipo de retorno

db2gse.ST_LineString

Exemplos

Um ornitologista, que está estudando a população de pássaros em várias ilhas marinhas do sul, sabe que a zona de alimentação de uma determinada espécie passiva restringe-se à costa. Algumas ilhas têm vários lagos. As praias dos lagos são habitadas exclusivamente por outras espécies mais agressivas. O ornitologista sabe que para cada ilha, se o perímetro do lago exceder um determinado limite, as espécies agressivas se tornarão mais numerosas e ameaçarão as espécies passivas do litoral. Portanto, ele solicita o perímetro agregado dos anéis internos das ilhas.

Na Figura 34, os anéis externos das ilhas representam a interface ecológica que cada ilha compartilha com o mar. Algumas ilhas têm lagos, que são representados pelos anéis interiores dos polígonos.

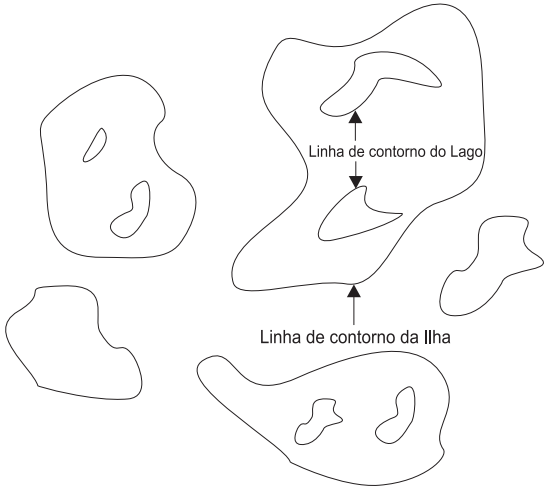


Figura 34. Usando ST_InteriorRingN para determinar o comprimento das margens dos lagos dentro de cada ilha

As colunas ID e NAME da tabela ISLANDS identificam cada ilha, ao passo que a coluna polígono de terra armazena a geometria da ilha.

```
CREATE TABLE ISLANDS (id    integer,
                      name  varchar(32),
                      land  db2gse.ST_Polygon);
```

O seguinte programa ODBC utiliza a função ST_InteriorRingN para extrair o anel interior (lago) de cada polígono da ilha como uma cadeia de linhas. O perímetro da cadeia de linhas que a função comprimento retorna é somado e exibido junto à ID da ilha.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include "sg.h"
#include "sgerr.h"
#include "sqlcli1.h"

/***                          ***
 *** Altere estas constantes ***
 ***                          ***/

#define USER_NAME   "sdetest"   /* nome de usuário */
#define USER_PASS   "acid.rain" /* sua senha de usuário */
#define DB_NAME     "mydb"      /* banco de dados ao qual se conectar */

static void check_sql_err (SQLHDBC  handle,
                           SQLHSTMT hstmt,
  LONG          rc,
                           CHAR     *str);

void main( argc, argv )
int argc;
char *argv[];
{
  SQLHDBC       handle;
  SQLHENV       henv;
  CHAR          sql_stmt[256];
  LONG          rc,
                total_perimeter,
                num_lakes,
                lake_number,
                island_id,
                lake_perimeter;
  SQLHSTMT      island_cursor,
                lake_cursor;
  SDWORD        pcbvalue,
                id_ind,
                lake_ind,
                length_ind;

/* Alocar memória para a manipulação de ambiente ODBC henv e
```

```
inicializar a aplicação. */

  rc = SQLAllocEnv (&henv);
    if (rc != SQL_SUCCESS)
  {
    printf ("SQLAllocEnv failed with %d\n", rc);
    exit(0);
  }

/* Alocar memória para o manipulador da conexão dentro do ambiente henv. */

  rc = SQLAllocConnect (henv, &handle);
    if (rc != SQL_SUCCESS)
  {
    falha de printf ("SQLAllocConnect com %d\n", rc);
    exit(0);
  }

/* Carregar o controlador ODBC e conectar-se à fonte de dados
   identificada pelo banco de dados, usuário e senha.*/

  rc = SQLConnect (handle,
                   (UCHAR *)DB_NAME,
                   SQL_NTS,
                   (UCHAR *)USER_NAME,
                   SQL_NTS,
                   (UCHAR *)USER_PASS,
                   SQL_NTS);

  check_sql_err (handle, NULL, rc, "SQLConnect");

/* Alocar memória para a instrução SQL manipular island_cursor. */

  rc = SQLAllocStmt (handle, &island_cursor);
  check_sql_err (handle, NULL, rc, "SQLAllocStmt");

/* Preparar e executar a consulta para obter as IDs e número da ilha de
   lagos (anéis interiores) */

  strcpy (sql_stmt, "select id, db2gse.ST_NumInteriorRings(land) from ISLANDS");

  rc = SQLExecDirect (island_cursor, (UCHAR *)sql_stmt, SQL_NTS);
  check_sql_err (NULL, island_cursor, rc, "SQLExecDirect");

/* Fazer o bind da coluna ID da tabela island à variável island_id */

  rc = SQLBindCol (island_cursor, 1, SQL_C_SLONG, &island_id, 0, &id_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

/* Fazer o bind do resultado de numinteriorrings(land) à variável num_lakes. */

  rc = SQLBindCol (island_cursor, 2, SQL_C_SLONG, &num_lakes, 0, &lake_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

/* Alocar memória para o manipulador da instrução SQL lago_cursor. */
```

```
rc = SQLAllocStmt (handle, &lake_cursor);
  check_sql_err (handle, NULL, rc, "SQLAllocStmt");

/* Preparar a consulta para obter o comprimento de um anel interior.*/

  strcpy (sql_stmt,
          "select Length(db2gse.ST_InteriorRingN(land, cast (? as
integer))) from ISLANDS where id = ?");

  rc = SQLPrepare (lake_cursor, (UCHAR *)sql_stmt, SQL_NTS);
  check_sql_err (NULL, lake_cursor, rc, "SQLPrepare");

/* Fazer o bind da variável lake_number como primeiro parâmetro de entrada */

  pcbvalue = 0;
  rc = SQLBindParameter (lake_cursor, 1, SQL_PARAM_INPUT, SQL_C_LONG,
       SQL_INTEGER, 0, 0, &lake_number, 0, &pcbvalue);
  check_sql_err (NULL, lake_cursor, rc, "SQLBindParameter");

/* Fazer o bind da id da ilha como segundo parâmetro de entrada */

  pcbvalue = 0;
  rc = SQLBindParameter (lake_cursor, 2, SQL_PARAM_INPUT, SQL_C_LONG,
       SQL_INTEGER, 0, 0, &island_id, 0, &pcbvalue);
  check_sql_err (NULL, lake_cursor, rc, "SQLBindParameter");

/* Fazer o bind do resultado de Length(db2gse.ST_InteriorRingN(land, cast
  (? como inteiro ))) à variável lake_perimeter */

  rc = SQLBindCol (lake_cursor, 1, SQL_C_SLONG, &lake_perimeter, 0,
                   &length_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

  /* Loop externo, obter as ids da ilha e número de lagos
  (anéis interiores) */

  while (SQL_SUCCESS == rc)
  {
    /* Buscar uma ilha */

    rc = SQLFetch (island_cursor);

    if (rc != SQL_NO_DATA)
    {
      check_sql_err (NULL, island_cursor, rc, "SQLFetch");

      /* Loop interno, desta ilha, obter o perímetro de todos os
         seus lagos (anéis internos) */

      para (perímetro_total = 0,lake_number = 1;
           lake_number <= num_lakes;
           lake_number++)
      {
        rc = SQLExecute (lake_cursor);
```

```
        check_sql_err (NULL, lake_cursor, rc, "SQLExecute");

        rc = SQLFetch (lake_cursor);
        check_sql_err (NULL, lake_cursor, rc, "SQLFetch");

        total_perimeter += lake_perimeter;

        SQLFreeStmt (lake_cursor, SQL_CLOSE);
      }
    }

/* Exibir a id da ilha e o perímetro total de seus lagos. */

    printf ("Island ID = %d,  Total lake perimeter = %d\n",
             island_id,total_perimeter);

  }

  SQLFreeStmt (lake_cursor, SQL_DROP);
  SQLFreeStmt (island_cursor, SQL_DROP);
        SQLDisconnect (handle);
        SQLFreeConnect (handle);
  SQLFreeEnv (henv);

  printf( "\nTest Complete ...\n" );

}

static void check_sql_err (SQLHDBC handle, SQLHSTMT  hstmt, LONG rc,
CHAR     *str)
{

    SDWORD dbms_err = 0;
    SWORD  length;
    UCHAR   err_msg[SQL_MAX_MESSAGE_LENGTH], state[6];

    if (rc != SQL_SUCCESS)
    {
      SQLError (SQL_NULL_HENV, handle, hstmt, state, &dbms_err,
                err_msg, SQL_MAX_MESSAGE_LENGTH - 1, &length);
      printf ("%s ERROR (%d): DBMS code:%d, SQL state: %s, message:
               \n %s\n", str, rc, dbms_err, state, err_msg);

      if (handle)
      {
        SQLDisconnect (handle);
        SQLFreeConnect (handle);
      }
      exit(1);
    }
}
```

ST_Intersection

ST_Intersection toma dois objetos de geometria e retorna o conjunto de interseção como um objeto de geometria.

Sintaxe

db2gse.ST_Intersection(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

O oficial encarregado do serviço contra incêndios deve obter as áreas de hospitais, escolas e casas de repouso que fazem interseção com o raio de uma possível contaminação por lixo tóxico.

As áreas sensíveis são armazenadas na tabela SENSITIVE_AREAS que é criada com a seguinte instrução CREATE TABLE. A coluna ZONE é definida como um polígono, que armazena a descrição de cada área sensível.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

As áreas de risco são armazenadas na tabela HAZARDOUS_SITES que foi criada com a seguinte instrução CREATE TABLE. A coluna LOCATION, definida como um ponto, armazena uma localização que é o centro geográfico de cada local arriscado.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

A função de buffer gera um buffer de 8 km ao redor das áreas do lixo tóxico. A função ST_Intersection gera polígonos da interseção dos polígonos de locais de lixo tóxico em buffer e das áreas sensíveis. A função ST_Area retorna a área de polígono da interseção, que é resumida para cada local de risco pela função SUM. A cláusula GROUP BY direciona a consulta para agregar as áreas de interseção pela ID do local do lixo tóxico.

```
SELECT hs.name,SUM(db2gse.ST_Area(db2gse.ST_Intersection (sa.zone,
db2gse.ST_buffer hs.location,(5 * 5280))))
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
GROUP BY hs.site_id;
```

Na Figura 35 na página 230, os círculos representam os polígonos em buffer ao redor de locais de lixo tóxico. A interseção destes polígonos em buffer com os polígonos da área sensível produz três outros polígonos. O hospital no canto

superior esquerdo faz interseção duas vezes, ao passo que a escolha no canto inferior direito faz apenas uma vez.

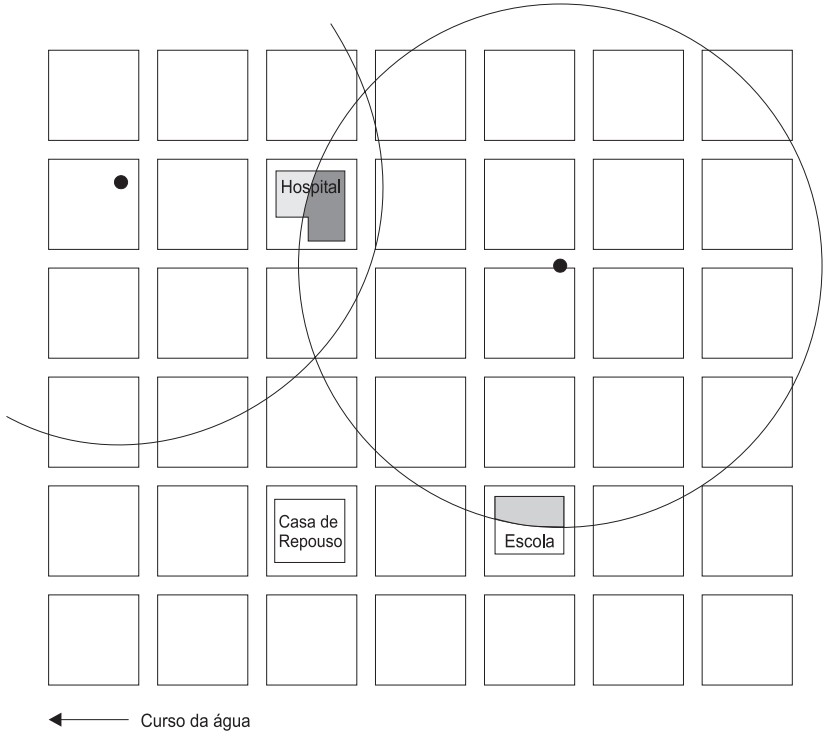


Figura 35. Utilizando ST_Intersection para determinar o quanto uma área em cada construção poderá ser afetada pelo lixo tóxico

ST_Intersects

ST_Intersects toma duas geometrias e retorna 1 (VERDADEIRO), se a interseção das duas geometrias não resultar num conjunto vazio. Do contrário, retornará 0 (FALSO).

Sintaxe

db2gse.ST_Intersects(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

O oficial encarregado do serviço contra incêndios precisa de uma lista de todas as áreas sensíveis dentro de de um raio de 8 km de uma área de lixo tóxico.

As áreas sensíveis são armazenadas na tabela SENSITIVE_AREAS que é criada com a seguinte instrução CREATE TABLE. A coluna ZONE é definida como um polígono, que armazena a descrição de cada área sensível.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

As áreas de risco são armazenadas na tabela HAZARDOUS_SITES criada com a seguinte instrução CREATE TABLE. A coluna LOCATION, definida como um ponto, armazena uma localização que é o centro geográfico de cada local arriscado.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

A consulta retorna uma lista de áreas sensíveis e nomes de áreas de risco para as áreas sensíveis que fazem interseção com o buffer de 8 km dos locais de risco.

```
SELECT sa.name, hs.name
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Intersects(db2gse.ST_Buffer(hs.location,(5 * 5280)),sa.zone)
=1;
```

ST_IsClosed

ST_IsClosed toma uma cadeia de linhas ou cadeia de linhas múltiplas e retorna 1 (VERDADEIRO) se estiver fechado; do contrário, retorna 0 (FALSO).

Sintaxe

db2gse.ST_IsClosed(c db2gse.ST_Curve)
db2gse.ST_IsClosed(mc db2gse.ST_MultiCurve)

Tipo de retorno

Inteiro

Exemplos

A seguinte instrução CREATE TABLE cria a tabela CLOSED_LINESTRING, que tem uma coluna de cadeia de linhas simples.

```
CREATE TABLE CLOSED_LINESTRING (ln1 db2gse.ST_LineString)
```

A seguinte instrução INSERT insere dois registros na tabela CLOSED_LINESTRING. O primeiro registro não é uma cadeia de linha fechada, ao passo que o segundo sim.

```
INSERT INTO CLOSED_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO CLOSED_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring  (10.02 20.01,11.92 35.64,
       25.02 34.15, 19.15 33.94,10.02 20.01)',
                     db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostra os resultados da função ST_IsClosed. A primeira linha retorna 0 porque a cadeia de linha não está fechada, ao passo que a segunda linha retorna 1 porque a cadeia de linha está fechada.

```
SELECT db2gse.ST_IsClosed(ln1) "Is it closed" FROM CLOSED_LINESTRING

Is it closed
------------
0
1

  2 record(s) selected.
```

A seguinte instrução CREATE TABLE cria a tabela CLOSED_MULTILINESTRING, que tem uma coluna de cadeia de linhas múltiplas.

```
CREATE TABLE CLOSED_MULTILINESTRING (mln1 db2gse.ST_MultiLineString)
```

A seguinte instrução INSERT insere dois registros em CLOSED_MULTILINESTRING, um registro de cadeia de linhas múltiplas não está fechado e outro está.

```
INSERT INTO CLOSED_MULTILINESTRING
VALUES(db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
       11.92 25.64), (9.55 23.75,15.36 30.11))',
                      db2gse.coordref()..srid(0)))

INSERT INTO CLOSED_MULTILINESTRING
VALUES(db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,11.92 35.64,
       25.02 34.15, 19.15 33.94,10.02 20.01),
                                        (51.71 21.73,73.36 27.04,71.52 32.87,
                                         52.43 31.90,51.71 21.73))',
                      db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostra os resultados da função ST_IsClosed. A primeira linha retorna 0 porque a cadeia de linhas múltiplas não está fechada, ao passo que a segunda linha retorna 1 porque a cadeia de linhas múltiplas está fechada. Uma cadeia de linhas múltiplas será fechada se todos os elementos da cadeia de linhas estiverem fechados.

```
SELECT db2gse.ST_IsClosed(mln1) "Is it closed" FROM CLOSED_MULTILINESTRING

Is it closed
------------
0
1

  2 record(s) selected.
```

ST_IsEmpty

ST_IsEmpty toma um objeto de geometria e retorna 1 (VERDADEIRO) se estiver vazio; do contrário, retornará 0 (FALSO).

Sintaxe

db2gse.ST_IsEmpty(g db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A seguinte instrução CREATE TABLE cria a tabela EMPTY_TEST, com duas colunas. A coluna GEOTYPE armazena o tipo de dados das classes filhas que estão armazenadas na coluna de geometria G1.

```
CREATE TABLE EMPTY_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

A seguinte instrução INSERT insere dois registros para o ponto, cadeia de linha e polígono da classes filhas de geometria. Um registro está vazio e outro não.

```
INSERT INTO EMPTY_TEST
VALUES('Point', db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Point', db2gse.ST_PointFromText('point empty',
db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Linestring', db2gse.ST_LineFromText('linestring (10.02 20.01,
       10.32 23.98, 11.92 25.64)',
                                db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Linestring', db2gse.ST_LineFromText('linestring  empty',
db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Polygon', db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,
25.02 34.15,19.15 33.94,10.02 20.01))',
                             db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Polygon', db2gse.ST_PolyFromText('polygon  empty',
db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostram a coluna tipo de geometria e os resultados da função ST_IsEmpty.

```
SELECT geotype, db2gse.ST_IsEmpty(g1) "It is empty" FROM EMPTY_TEST

GEOTYPE                  It is empty
```

```
--------------------      -----------
ST_Point                        0
ST_Point                        1
ST_Linestring                   0
ST_Linestring                   1
ST_Polygon                      0
ST_Polygon                      1

  6 record(s) selected.
```

ST_IsRing

ST_IsRing toma uma cadeia de linhas e retorna 1 (VERDADEIRO) se for um anel (a saber, a cadeia de linhas é fechada e simples); caso contrário, retornará 0 (FALSO).

Sintaxe

db2gse.ST_IsRing(c db2gse.ST_Curve)

Tipo de retorno

Inteiro

Exemplos

A seguinte instrução CREATE TABLE cria a tabela RING_LINESTRING, que tem uma coluna de cadeia de linhas simples denominada LN1.

```
CREATE TABLE RING_LINESTRING (ln1 db2gse.ST_LineString)
```

As seguintes instruções INSERT inserem três cadeias de linhas na coluna LN1. A primeira linha contém uma cadeia de linha que não é fechada e portanto não é um anel. A segunda linha contém uma cadeia de linhas que é fechada e é simples, portanto, é um anel. A terceira linha contém uma cadeia de linhas que está fechada mas não é simples porque faz interseção com seu próprio interior e, portanto, não é um anel.

```
INSERT INTO RING_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO RING_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring (10.02 20.01,11.92 35.64,25.02 34.15,
                                19.15 33.94, 10.02 20.01)',
                    db2gse.coordref()..srid(0)))

INSERT INTO RING_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring (15.47 30.12,20.73 22.12,10.83 14.13,
                                16.45 17.24,21.56 13.37,11.23 22.56,
                                19.11 26.78,15.47 30.12)',
                    db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostra os resultados da função ST_IsRing. A primeira e a terceira linha retornam 0. Isto porque as cadeias de linhas não são anéis, ao passo que a segunda linha retorna 1 porque é um anel.

```
SELECT db2gse.ST_IsRing(ln1) "Is it ring" FROM RING_LINESTRING

Is it ring
-----------
          0
```

```
          1
          0

  3 record(s) selected.
```

ST_IsSimple

ST_IsSimple toma um objeto de geometria e retorna 1 (VERDADEIRO) se o objeto for simples; do contrário, retornará 0 (FALSO).

Sintaxe

db2gse.ST_IsSimple(g db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A seguinte instrução CREATE TABLE cria a tabela ISSIMPLE_TEST, a qual possui duas colunas. A coluna PID, que é smallint, contém o identificador exclusivo para cada linha. A coluna de geometria G1 armazena as amostras de geometria simples e não-simples.

```
CREATE TABLE ISSIMPLE_TEST (pid smallint, g1 db2gse.ST_Geometry)
```

A seguinte instrução INSERT insere dois registros na tabela ISSIMPLE_TEST. O primeiro é simples porque é uma cadeia de linha que não faz interseção com seu interior. O segundo é não-simples porque faz interseção com seu interior.

```
INSERT INTO ISSIMPLE_TEST
VALUES (1, db2gse.ST_LineFromText('linestring (10 10, 20 20, 30 30)',
db2gse.coordref()..srid(0)))

INSERT INTO ISSIMPLE_TEST
VALUES (2, db2gse.ST_LineFromText('linestring (10 10, 20 20,20 30,10 30,10 20,
20 10)', db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostram os resultados da função ST_IsSimple. O primeiro registro retorna 1 porque a cadeia de linha é simples, ao passo que o segundo registro retorna 0 porque a cadeia de linha não é simples.

```
SELECT ST_IsSimple(g1)
FROM ISSIMPLE_TEST

g1
--------------
1
0
```

ST_IsValid

ST_IsValid toma um ST_Geometry e retorna 1 (VERDADEIRO) se for válido, do contrário, retornará 0 (FALSO). Uma geometria inserida num banco de dados do DB2 sempre será válida, porque DB2 Spatial Extender sempre valida seus dados espaciais antes de aceitá-los. No entanto, outros fornecedores DBMS poderão não validar a entrada, mas sim solicitar que essa aplicação o faça.

Sintaxe

db2gse.ST_IsValid(g db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A tabela valid_test é criada com as colunas geotype e g1. A coluna geotype armazena o nome da subclasse de geometria armazenada na coluna de geometria g1.

```
CREATE TABLE valid_test (geotype varchar(20), g1 db2gse.ST_Geometry)
```

As instruções INSERT inserem uma subclasse de amostra na tabela valid_test.

```
INSERT INTO valid_test VALUES(
    'Point', db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
       11.92 25.64)',
db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Polygon', db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,
    25.02 34.15, 19.15 33.94,10.02 20.01))', db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Multipoint', db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Multilinestring', db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,
10.32 23.98,11.92 25.64),(9.55 23.75,15.36 30.11))',
db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Multipolygon',
```

```
db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
25.02 34.15,19.15 33.94,10.02 20.01)),((51.71 21.73,73.36 27.04,71.52 32.87,
52.43 31.90,51.71 21.73)))', db2gse.coordref()..srid(0))
)
```

A instrução SELECT relaciona o nome da subclasse armazenado na coluna geotype à dimensão desse geotipo.

```
SELECT geotype, db2gse.ST_IsValid(g1) Valid FROM valid_test

GEOTYPE              Valid
-------------------- -------------
ST_Point                     1
ST_Linestring                1
ST_Polygon                   1
ST_Multipoint                1
ST_Multilinestring           1
ST_Multipolygon              1

6 record(s) selected.
```

ST_Length

ST_Length toma uma cadeia de linhas ou cadeia de linhas múltiplas e retorna seu comprimento.

Sintaxe

db2gse.ST_Length(c db2gse.ST_Curve)
db2gse.ST_Length(mc db2gse.ST_MultiCurve)

Tipo de retorno

Duplo

Exemplos

Um ecologista local está estudando os padrões migratórios da população de salmão nos cursos de água do município. O ecologista deseja obter o comprimento de todo o sistema fluvial que passa pelo município.

A seguinte instrução CREATE TABLE cria a tabela WATERWAYS. As colunas ID e NAME identificam cada sistema fluvial que está armazenado na tabela. A coluna ÁGUA é uma cadeia de linhas múltiplas pois os sistemas de rios e córregos são geralmente uma agregação de várias cadeias de linhas.

```
CREATE TABLE WATERWAYS (id integer, name varchar(128),
water db2gse.ST_MultiLineString);
```

A seguinte instrução SELECT utiliza a função ST_Length para retornar o nome e o comprimento de cada curso de água dentro do município.

```
SELECT name, db2gse.ST_Length(water) "Length"
FROM WATERWAYS;
```

a Figura 36 na página 242 exibe os sistemas fluviais que estão dentro dos limites do município.

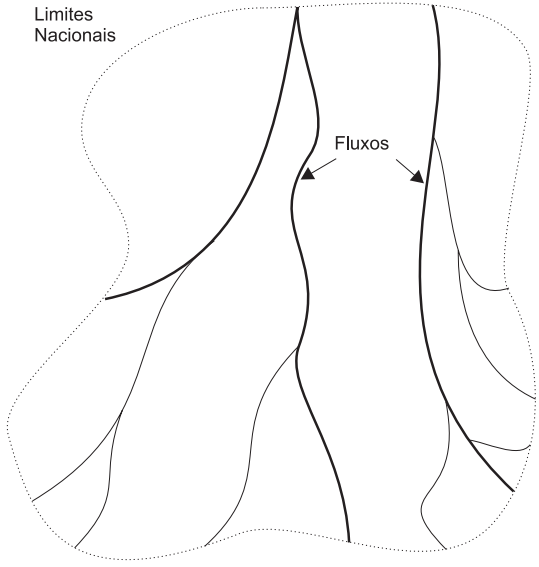


Figura 36. Utilizando ST_Length para determinar o comprimento total dos cursos de água num município

ST_LineFromText

ST_LineFromText toma uma representação de texto reconhecido do tipo cadeia de linha e uma identidade do sistema de referência espacial e retorna uma cadeia de linha.

Sintaxe

db2gse.ST_LineFromText(lineStringTaggedText Varchar(4000), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_LineString

Exemplos

A seguinte instrução CREATE TABLE cria a tabela LINESTRING_TEST, que tem uma coluna de cadeia de linhas simples LN1.

```
CREATE TABLE LINESTRING_TEST (ln1 db2gse.ST_LineString)
```

A seguinte instrução INSERT insere uma cadeia de linhas na coluna LN1 usando a função ST_LineFromText.

```
INSERT INTO LINESTRING_TEST
VALUES (db2gse.ST_LineFromText('linestring(10.01 20.03,20.94 21.34,
        35.93 19.04)', db2gse.coordref()..srid(0)))
```

ST_LineFromWKB

ST_LineFromWKB toma uma representação do modo binário reconhecida do tipo cadeia de linha e uma identidade do sistema de referência espacial e retorna uma cadeia de linha.

Sintaxe

db2gse.ST_LineFromWKB(WKBLineString Blob(1M), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_LineString

Exemplos

O seguinte fragmento de código preenche a tabela SEWERLINES com id, classe de tamanho e geomeria exclusivos de cada linha sewer.

A tabela SEWERLINES foi criada com três colunas. A primeira coluna, SEWER_ID, identifica exclusivamente cada linha do sewer. A segunda coluna, CLASS, do tipo inteiro identifica o tipo de linha de sewer, que geralmente está associada à capacidade da linha. A terceira coluna, SEWER, do tipo cadeia de linhas armazena a geometria da linha sewer.

```
CREATE TABLE SEWERLINES (sewer_id   integer,
                         class  integer,
                         sewer  db2gse.ST_LineString);

/* Criar a instrução insert do SQL para preencher a id do sewer, o tamanho
   e a cadeia de linha sewer. Os pontos de interrogação são marcadores de
   parâmetros que indicam a id do sewer, a classe e os valores de
   geometria do sewer que serão recuperados no tempo de execução. */
strcpy (wkb_sql,"insert into sewerlines (sewer_id,class,sewer)
values (?,?, db2gse.ST_LineFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da id do sewer ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_id, 0, &pcbvalue1);

/* Fazer o bind do valor da classe de inteiro ao segundo parâmetro. */
pcbvalue2 = 0;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_class, 0, &pcbvalue2);

/* Fazer o bind do shape ao terceiro parâmetro. */
pcbvalue3 = blob_len;
```

```
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, sewer_wkb, blob_len, &pcbvalue3);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

ST_MLineFromText

ST_MLineFromText toma uma representação de texto reconhecida do tipo cadeia de linhas múltiplas e uma identidade do sistema de referência espacial e retorna uma cadeia de linhas múltiplas.

Sintaxe

db2gse.ST_MLineFromText(multiLineStringTaggedText String, cr db2gse.coordref)

Tipo de retorno

db2gse.ST_MultiLineString

Exemplos

A seguinte instrução CREATE TABLE cria a tabela MLINESTRING_TEST. MLINESTRING_TEST tem duas colunas: a coluna GID smallint, que identifica exclusivamente a linha e a coluna de cadeia de linhas múltiplas ML1.

```
CREATE TABLE ST_MLINESTRING_TEST (gid smallint, ml1 db2gse.ST_MultiLineString)
```

A seguinte instrução INSERT insere a cadeia de linhas múltiplas com a função ST_MLineFromText.

```
INSERT INTO MLINESTRING_TEST
VALUES (1, db2gse.ST_MLineFromText('multilinestring((10.01 20.03,10.52 40.11,
         30.29 41.56,31.78 10.74),
                                        (20.93 20.81, 21.52 40.10))',
                          db2gse.coordref()..srid(0)))
```

ST_MLineFromWKB

ST_MLineFromWKB toma uma representação de modo binário reconhecida do tipo cadeia de linhas múltiplas e uma identidade do sistema de referência espacial e retorna uma cadeia de linhas múltiplas.

Sintaxe

db2gse.ST_MLineFromWKB(WKBMultiLineString Blob(1M), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_MultiLineString

Exemplos

O seguinte fragmento de código preenche a tabela WATERWAYS com uma id exclusiva, um nome e uma cadeia de linhas múltiplas de água.

A tabela WATERWAYS é criada com as colunas ID e NAME que identificam cada sistema de córregos e rios armazenados na tabela. A coluna ÁGUA é uma cadeia de linhas múltiplas pois os sistemas de rios e córregos são geralmente uma agregação de várias cadeias de linhas.

```
CREATE TABLE WATERWAYS (id        integer,
                        name      varchar(128),
                        water     db2gse.ST_MultiLineString);

/* Criar a instrução insert do SQL para preencher a id, nome e
   a cadeia de linhas múltiplas. Os pontos de interrogação são
   marcadores de parâmetros que
   indicam a id, o nome e os valores da água que serão recuperados no
 tempo de execução.*/
strcpy (shp_sql,"insert into WATERWAYS (id,name,water)
values (?,?, db2gse.ST_MLineFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da id ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);

/* Fazer o bind do valor do nome varchar ao segundo parâmetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, name_len, 0, &name, name_len, &pcbvalue2);

/* Fazer o bind do shape ao terceiro parâmetro. */
```

```
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, water_shape, blob_len, &pcbvalue3);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

ST_MPointFromText

ST_MPointFromText toma uma representação de texto reconhecida do tipo multiponto e uma identidade do sistema de referência espacial e retorna um multiponto.

Sintaxe

db2gse.ST_MPointFromText(multiPointTaggedText Varchar(4000), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_MultiPoint

Exemplos

A seguinte instrução CREATE TABLE cria a tabela MULTIPOINT_TEST com uma coluna de multiponto simples, MPT1.

```
CREATE TABLE MULTIPOINT_TEST (mpt1 db2gse.ST_MultiPoint)
```

A seguinte instrução INSERT insere um multiponto na coluna MPT1 utilizando a coluna ST_MPointFromText.

```
INSERT INTO MULTIPOINT_TEST
VALUES (1, db2gse.ST_MPointFromText('multipoint(10.01 20.03,10.52 40.11,
30.29 41.56,31.78 10.74)',                          db2gse.coordref()..srid(0)))
```

ST_MPointFromWKB

ST_MPointFromWKB toma uma representação de modo binário reconhecida do tipo multiponto e uma identidade do sistema de referência espacial para retornar um multiponto.

Sintaxe

db2gse.ST_MPointFromWKB(WKBMultiPoint Blob(1M), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_MultiPoint

Exemplos

O seguinte fragmento de código preencha a tabela SPECIES_SITINGS.

A tabela SPECIES_SITINGS foi criada com três colunas. As colunas SPECIES e GENUS identificam exclusivamente cada linha, os passos que o multiponto SITTINGS armazena as localizações dos sítios das espécies.

```
CREATE TABLE SPECIES_SITINGS (species  varchar(32),
                              genus varchar(32),
                              sitings  db2gse.ST_MultiPoint);

/* Criar a instrução insert do SQL para preencher as espécies, gêneros e
   sítios. Os pontos de interrogação são marcadores de parâmetros que
   indicam as espécies, os gêneros e os valores do sítio que serão recuperados no
 tempo de execução.*/
strcpy (wkb_sql,"insert into SPECIES_SITINGS (species,genus,sitings)
values (?,?, db2gse.ST_MPointFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Fazer o bind do valor de espécies varchar ao primeiro parâmetro. */
pcbvalue1 = species_len;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, species_len, 0, &species, species_len, &pcbvalue1);
/* Fazer o bind do valor do gênero varchar ao segundo parâmetro. */
pcbvalue2 = genus_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, genus_len, 0, &name, genus_len, &pcbvalue2);

/* Fazer o bind do shape ao terceiro parâmetro. */
pcbvalue3 = sitings_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, sitings_len, 0, sitings_wkb, sitings_len, &pcbvalue3);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

ST_MPolyFromText

ST_MPolyFromText toma uma representação de texto reconhecida do tipo multipolígono e uma identidade do sistema de referência espacial e retorna um multipolígono.

Sintaxe

db2gse.ST_MPolyFromText(multiPolygonTaggedText Varchar(4000), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_MultiPolygon

Exemplos

A seguinte instrução CREATE TABLE cria a tabela MULTIPOLYGON_TEST, que tem uma coluna de multipolígono simples, MPL1.

```
CREATE TABLE MULTIPOLYGON_TEST (mpl1 db2gse.ST_MultiPolygon)
```

A seguinte instrução INSERT insere um multipolígono na coluna MPL1 usando a função ST_MPolyFromText.

```
INSERT INTO MULTIPOLYGON_TEST VALUES (
db2gse.ST_MPolyFromText('multipolygon(((10.01 20.03,10.52 40.11,
 30.29 41.56,31.78 10.74,10.01 20.03),(21.23 15.74,21.34 35.21,28.94 35.35,
 29.02 16.83, 21.23 15.74)),((40.91 10.92,40.56 20.19,
 50.01 21.12,51.34 9.81, 40.91 10.92)))',
 db2gse.coordref()..srid(0)))
```

ST_MPolyFromWKB

ST_MPolyFromWKB toma uma representação de modo binário reconhecida do tipo multipolígono e uma identidade do sistema de referência espacial e retorna um multipolígono.

Sintaxe

db2gse.ST_MPolyFromWKB(WKBMultiPolygon Blob(1M), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_MultiPolygon

Exemplos

O seguinte fragmento de código preenche a tabela LOTES.

A tabela LOTS armazena o LOT_ID que identifica exclusivamente cada lote e o multipolígono do LOT que contém a geometria de linha do lote.

```
CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );

/* Criar a instrução insert do SQL para preencher a id do lote e o lote. Os
   pontos de interrogação são marcadores de parâmetro que indicam a id do
   lote e os valores do
   lote que serão recuperados no tempo de execução. */
strcpy (wkb_sql,"insert into LOTS (lot_id,lot)
values (?, db2gse.ST_MPolyFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da id do lote ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Fazer o bind do shape do lote ao segundo parâmetro. */
pcbvalue2 = lot_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, lot_len, 0, lot_wkb, lot_len, &pcbvalue2);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

ST_NumGeometries

ST_NumGeometries toma uma coleção e retorna o número de geometrias na coleção.

Sintaxe

db2gse.ST_NumGeometries(g db2gse.ST_GeomCollection)

Tipo de retorno

Inteiro

Exemplos

O engenheiro da cidade precisa saber o número real de construções do distrito associado a cada base da construção.

As bases da construção são armazenadas na tabela BUILDINGFOOTPRINTS que foi criada com a seguinte instrução CREATE TABLE.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

A seguinte instrução SELECT usa a função ST_NumGeometries para relacionar o BUILDING_ID que identifica exclusivamente cada construção e o número de construções em cada base.

```
SELECT building_id, db2gse.ST_NumGeometries (footprint) "Number of buildings"
FROM BUILDINGFOOTPRINTS;
```

ST_NumInteriorRing

ST_NumInteriorRing toma um polígono e retorna o número de seus anéis interiores.

Sintaxe

db2gse.NumInteriorRing(p db2gse.ST_Polygon)

Tipo de retorno

Inteiro

Exemplos

Uma ornitologista que deseja estudar a população de pássaros em várias ilhas marinhas do sul, sabe que a zona de alimentação de uma determinada espécie restringe-se às ilhas que contém os lagos de água fresca. Portanto, ela deseja saber quais ilhas contêm um ou mais lagos.

A seguinte instrução CREATE TABLE cria a tabela ISLANDS. As colunas ID e NAME da tabela ISLANDS identificam cada ilha e a coluna polígono de LAND armazena a geometria da ilha.

```
CREATE TABLE ISLANDS (id integer, name varchar(32), land db2gse.ST_Polygon);
```

Como os anéis interiores representam os lagos, a função ST_NumInteriorRing é usada para relacionar somente essas ilhas que têm pelo menos um anel interior.

```
SELECT name FROM ISLANDS WHERE db2gse.ST_NumInteriorRing(land) > 0;
```

ST_NumPoints

ST_NumPoints toma uma cadeia de linha e retorna seu número de pontos.

Sintaxe

db2gse.ST_NumPoints(l db2gse.ST_LineString)

Tipo de retorno

Inteiro

Exemplos

A seguinte instrução CREATE TABLE cria a tabela NUMPOINTS_TEST. A coluna GEOTYPE contém o tipo de geometria armazenado na coluna de geometria G1.

```
CREATE TABLE NUMPOINTS_TEST (geotype varchar(12), g1 db2gse.ST_Geometry)
```

A instrução INSERT abaixo insere uma cadeia de linhas.

```
INSERT INTO NUMPOINTS_TEST VALUES( linestring,
db2gse.ST_LineFromText('linestring (10.02 20.01, 23.73 21.92)',
db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente relaciona o tipo de geometria e o número de pontos contidos dentro de cada.

```
SELECT geotype, db2gse.ST_NumPoints(g1)
FROM NUMPOINTS_TEST

GEOTYPE      Number of points
------------ ----------------
ST_linestring          2
  1 record(s) selected.
```

ST_OrderingEquals

ST_OrderingEquals compara as duas geometrias e retorna 1 (VERDADEIRO) se as geometrias forem iguais e as coordenadas estiverem na mesma ordem; do contrário, retornará 0 (FALSO).

Sintaxe

db2gse.ST_OrderingEquals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

A seguinte instrução CREATE TABLE cria a tabela LINESTRING_TEST, que tem duas colunas de cadeia de linhas, L1 e L2.

```
CREATE TABLE LINESTRING_TEST (lid integer, l1 db2gse.ST_LineString,
l2 db2gse.ST_LineString);
```

A seguinte instrução INSERT insere duas cadeias de linha em L1 e L2 que são iguais e têm a mesma ordem de coordenadas.

```
INSERT INTO linestring_test VALUES (1,
db2gse.LineFromText('linestring (10.01 20.02, 21.50 12.10)',
  db2gse.coordref()..srid(0)),
db2gse.LineFromText('linestring (10.01 20.02, 21.50 12.10)',
db2gse.coordref()..srid(0)));
```

A seguinte instrução INSERT insere duas cadeias de linha em L1 e L2 que são iguais mas não têm a mesma ordem de coordenadas.

```
INSERT INTO linestring_test VALUES (2,
db2gse.LineFromText('linestring (10.01 20.02, 21.50 12.10)',
  db2gse.coordref()..srid(0)),
db2gse.LineFromText('linestring (21.50 12.10,10.01 20.02)',
db2gse.coordref()..srid(0)));
```

A seguinte instrução SELECT e o conjunto de resultado correspondente mostra como a função ST_Equals retorna 1 (VERDADEIRO), independente da ordem das coordenadas. A função ST_OrderingEquals retorna 0 (FALSO) se as geometrias não forem iguais e tiverem a mesma ordem de coordenadas.

```
SELECT lid, db2gse.ST_Equals(l1,l2) equals, db2gse.ST_OrderingEquals(l1,l2)
OrderingEquals
FROM linestring_test

lid   equals   OrderingEquals
---   ------   -----------
 1       1           1
 2       1           0
```

ST_Overlaps

ST_Overlaps toma dois objetos de geometria e retorna 1 (VERDADEIRO) se a interseção dos objetos resultar num objeto de geometria da mesma dimensão, mas não for igual a nenhum dos objetos de origem; do contrário, retornará 0 (FALSO).

Sintaxe

db2gse.ST_Overlaps(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

O supervisor do município precisa de uma lista de áreas de lixo tóxico cujo raio de 8 km abranja áreas sensíveis.

A seguinte instrução CREATE TABLE cria a tabela SENSITIVE_AREAS. A tabela SENSITIVE_AREAS contém várias colunas que descrevem as instituições ameaçadas além da coluna ZONE, que armazena a geometria do polígono da instituição.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

A tabela HAZARDOUS_SITES armazena a identidade dos locais nas colunas SITE_ID e NAME, ao passo que a localização geográfica real de cada local está armazenada na coluna de ponto LOCATION.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

Na seguinte instrução SELECT, as tabelas SENSITIVES_AREAS E HAZARDOUS_SITES são unidas pela função ST_Overlaps. Ela retorna 1 (VERDADEIRO) para todas as linhas na tabela SENSITIVE_AREAS cujo polígono da zona abrange o raio de 8 km de buffer do ponto de localização HAZARDOUS_SITES.

```
SELECT hs.name
FROM HAZARDOUS_SITES hs, SENSITIVE_AREAS sa
WHERE db2gse.ST_Overlaps (buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

Na Figura 37 na página 258, o hospital e a escola estão no raio de 8 km das áreas de lixo tóxico do município, ao passo que a casa de repouso está fora.

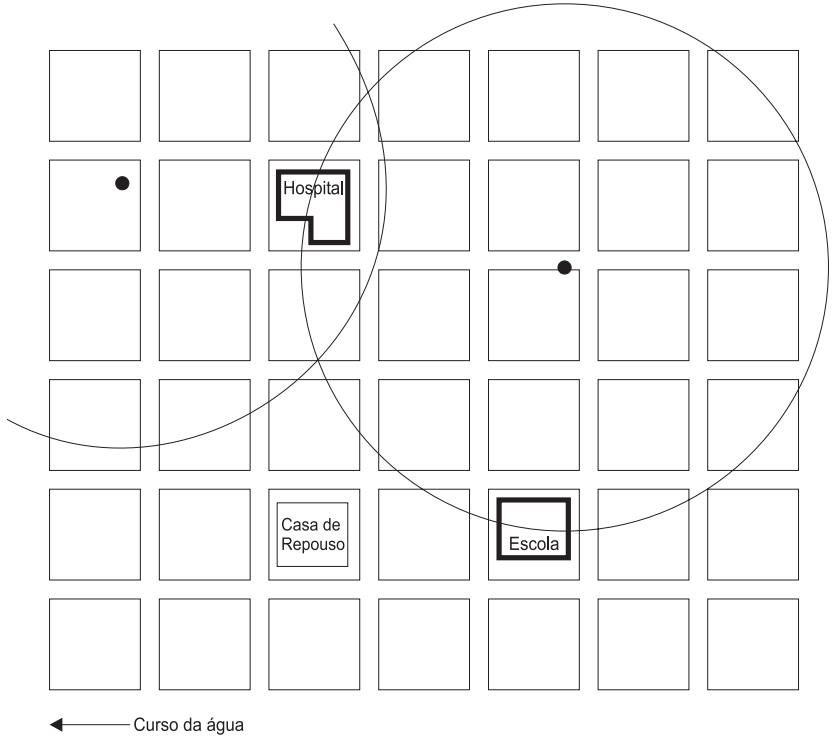


Figura 37. Usando ST_Overlaps para determinar as construções que pelo menos parcialmente estão dentro de uma área de lixo tóxico.

ST_Perimeter

ST_Perimeter retorna o perímetro de um ST_Surface.

Sintaxe

db2gse.ST_Perimeter(s db2gse.ST_Surface)
db2gse.ST_Perimeter(ms db2gse.ST_MultiSurface)

Tipo de retorno

Duplo

Exemplos

Um ecologista que estuda os pássaroes da costa para determinar a linha costeira dos lagos dentro de uma determinada área. Os lagos estão armazenados como multipolígonos na tabela WATERBODIES que foi criada com a seguinte instrução CREATE TABLE.

```
CREATE TABLE WATERBODIES (wbid  integer,
                          waterbody  db2gse.ST_MultiPolygon);
```

Na seguinte instrução SELECT, a função ST_Perimeter retorna o perímetro ao redor de cada extensão de água, ao passo que a função SUM agrega os perímetros para retornar seu total.

```
SELECT SUM(db2gse.ST_Perimeter(waterbody))
FROM waterbodies;
```

ST_PointFromText

ST_PointFromText toma uma representação de texto reconhecida do tipo ponto e uma identidade do sistema de referência espacial e retorna um ponto.

Sintaxe

db2gse.ST_PointFromText(pointTaggedText Varchar(4000), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_Point

Exemplos

A seguinte instrução CREATE TABLE cria a tabela POINT_TEST, que tem uma coluna de ponto simples, PT1.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

Antes de a instrução INSERT inserir o ponto na coluna PT1, a função ST_PointFromText converte as coordenadas de texto do ponto ao formato de ponto.

```
INSERT INTO POINT_TEST VALUES (
       db2gse.ST_PointFromText ('point(10.01 20.03)',

db2gse.coordref()..srid(0)))
```

ST_PointFromWKB

ST_PointFromWKB toma uma representação de modo binário reconhecida do tipo ponto e uma identidade do sistema de referência espacial para retornar um ponto.

Sintaxe

db2gse.ST_PointFromWKB(WKBPoint Blob(1M), srs SRID)

Tipo de retorno

db2gse.ST_Point

Exemplos

O seguinte fragmento de código preenche a tabela HAZARDOUS_SITES.

As áreas de risco são armazenadas na tabela HAZARDOUS_SITES criada com a seguinte instrução CREATE TABLE. A coluna LOCATION, definida como um ponto, armazena uma localização que é o centro geográfico de cada local arriscado.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);

/* Criar a instrução insert do SQL para preencher a id do local, nome e
   localização. Os pontos de interrogação são marcadores de parâmetros
   que indicam a
   id do local, nome e os valores de localização que serão recuperados no
   tempo de execução. */
strcpy (wkb_sql,"insert into HAZARDOUS_SITES (site_id, name, location)
values (?,?, db2gse.ST_PointFromWKB(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da id do local ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
   SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);

/* Fazer o bind do valor varchar do nome ao segundo parâmetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Fazer o bind do shape de localização ao terceiro parâmetro. */
pcbvalue3 = location_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
        SQL_BLOB, location_len, 0, location_wkb, location_len, &pcbvalue3);

    /* Executar a instrução insert . */
    rc = SQLExecute (hstmt);
```

ST_Point

ST_Point retorna um ST_Point, dada uma coordenada x, coordenada y e referência espacial.

Sintaxe

db2gse.ST_Point(X Double, Y Double, srs SRID)

Tipo de retorno

db2gse.ST_Point

Exemplos

A seguinte instrução CREATE TABLE cria a tabela POINT_TEST, que tem uma coluna de ponto simples, PT1.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

A função ST_Point converte as coordenadas de ponto em uma geometria de ponto antes de a instrução INSERT inseri-la na coluna PT1.

```
INSERT INTO point_test VALUES(
    db2gse.ST_Point(10.01,20.03, db2gse.coordref()..srid(0))
)
```

ST_PointN

ST_PointN toma uma cadeia de linha e um índice de inteiros e retorna um ponto que é o vértice nth no caminho da cadeia de linhas.

Sintaxe

db2gse.ST_PointN(l db2gse.ST_Curve, n Integer)

Tipo de retorno

db2gse.ST_Point

Exemplos

A seguinte tabela CREATE TABLE cria a tabela POINTN_TEST, que tem duas colunas: a coluna GID, que identifica exclusivamente cada linha, e a coluna da cadeia de linhas LN1.

```
CREATE TABLE POINTN_TEST (gid integer, ln1 db2gse.ST_LineString)
```

A seguinte instrução INSERT insere dois valores da cadeia de linhas. A primeira cadeia de linha não possui coordenadas Z ou medidas, ao passo que a segunda possui ambas.

```
INSERT INTO POINTN_TEST VALUES(1,
db2gse.ST_LineFromText('linestring (10.02 20.01,23.73 21.92,30.10 40.23)',
db2gse.coordref()..srid(0)))

INSERT INTO POINTN_TEST VALUES(2,
       db2gse.ST_LineFromText('linestring zm (10.02 20.01 5.0 7.0,
       23.73 21.92 6.5 7.1, 30.10 40.23 6.9 7.2)',
       db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente relacionam a coluna GID e o segundo vértice de cada cadeia de linha. A primeira linha resulta num ponto sem uma coordenada Z ou medida, ao passo que a segunda linha resulta num ponto com uma coordenada Z e uma medida. A função ST_PointN retorna pontos com uma coordenada Z ou uma medida, se existirem na cadeia de linhas de origem.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.ST_PointN(ln1,2)) AS varchar(60))
"The 2nd vertice"
FROM POINTN_TEST

GID         The 2nd vertice
----------- ------------------------------------------------------------
          1 POINT ( 23.73000000 21.92000000)
          2 POINT ZM ( 23.73000000 21.92000000 7.00000000 7.10000000)

  2 record(s) selected.
```

ST_PointOnSurface

O ST_PointOnSurface toma tanto um polígono como um multipolígono e retorna um ST_Point.

Sintaxe

db2gse.ST_PointOnSurface(s db2gse.ST_Surface)
db2gse.ST_PointOnSurface(ms db2gse.ST_MultiSurface)

Tipo de retorno

db2gse.ST_Point

Exemplos

O engenheiro da cidade precisa criar um ponto de rótulo para cada base de construção.

As bases da construção são armazenadas na tabela BUILDINGFOOTPRINTS que foi criada com a seguinte instrução CREATE TABLE.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

A função ST_PointOnSurface gera um ponto que tem garantia de estar na superfície das bases de construção. A função ST_PointOnSurface retorna um ponto que a função AsBinaryShape converte em um shape convertido em uma cadeia de caracteres de 1 megabyte para ser usada pela aplicação.

```
SELECT CAST(db2gse.AsBinaryShape(db2gse.ST_PointOnSurface(footprint)) as
            blob(1m))
FROM BUILDINGFOOTPRINTS;
```

ST_PolyFromText

ST_PolyFromText toma uma representação de texto reconhecida do tipo polígono e uma identidade do sistema de referência espacial e retorna um polígono.

Sintaxe

db2gse.ST_PolyFromText(polygonTaggedText Varchar(4000), cr db2gse.coordref)

Tipo de retorno

db2gse.ST_Polygon

Exemplos

A seguinte instrução CREATE TABLE cria a tabela POLYGON_TEST com a coluna de polígono simples.

```
CREATE TABLE POLYGON_TEST (pl1 db2gse.ST_Polygon)
```

A seguinte instrução INSERT insere um polígono na coluna de polígonos usando a função ST_PolyFromText.

```
INSERT INTO POLYGON_TEST VALUES (1,
db2gse.ST_PolyFromText('polygon((10.01 20.03,10.52 40.11,30.29 41.56,
    31.78 10.74,10.01 20.03))',

db2gse.coordref()..srid(0)))
```

ST_PolyFromWKB

ST_PolyFromWKB toma uma representação de modo binário reconhecida do tipo polígono e uma identidade do sistema de referência espacial para retornar um polígono.

Sintaxe

db2gse.ST_PolyFromWKB(WKBPolygon Blob(1M), SRID Integer)

Tipo de retorno

db2gse.ST_Polygon

Exemplos

O seguinte fragmento de código preenche a tabela SENSITIVE_AREAS.

A tabela SENSITIVE_AREAS contém várias colunas que descrevem as instituições ameaçadas além da coluna zona, que armazena a geometria do polígono da instituição.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);


/* Criar a instrução insert do SQL para preencher a id, nome, tamanho, tipo e
   zona. Os pontos de interrogação são marcadores de parâmetro que
   indicam a id, nome, tamanho, tipo e valores da zona que serão
   recuperados no tempo de execução. */
strcpy (shp_wkb,"insert into SENSITIVE_AREAS (id, name, size, type, zone)
values (?,?,?,?, db2gse.ST_PolyFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da id ao primeiro parâmetro.*/
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);

/* Fazer o bind do valor varchar do nome ao segundo parâmetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Fazer o bind da dimensão flutuante ao terceiro parâmetro. */
pcbvalue3 = 0;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_FLOAT,
```

```
     SQL_REAL, 0, 0, &size, 0, &pcbvalue3);

/* Fazer o bind do tipo varchar ao quarto parâmetro. */
pcbvalue4 = type_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_VARCHAR, type_len, 0, type, type_len, &pcbvalue4);

/* Fazer o bind do polígono de zona ao quinto parâmetro. */
pcbvalue5 = zone_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, zone_len, 0, zone_wkb, zone_len, &pcbvalue5);

/* Executar a instrução insert . */
rc = SQLExecute (hstmt);
```

ST_Polygon

O ST_Polygon gera um ST_Polygon a partir de um ST_LineString e o identificador de um sistema de referência espacial.

Sintaxe

db2gse.ST_Polygon(l db2gse.ST_LineString, cr db2gse.coordref)

Tipo de retorno

db2gse.ST_Polygon

Exemplos

A seguinte instrução CREATE TABLE cria a tabela POLYGON_TEST, que tem uma coluna simples, P1.

```
CREATE TABLE POLYGON_TEST (p1 db2gse.ST_polygon)
```

A seguinte instrução INSERT converte um anel (uma cadeia de linhas que é fechada e simples) em um polígono e insere-o na coluna P1 usando a função ST_LineFromText dentro da função ST_Polygon.

```
INSERT INTO POLYGON_TEST VALUES (
db2gse.ST_Polygon(db2gse.ST_LineFromText('linestring(10.01 20.03,20.94
21.34,35.93 10.04,10.01 20.03)', db2gse.coordref()..srid(0))),
db2gse.coordref()..srid(0)))
)
```

ST_Relate

ST_Relate compara duas geometrias e retorna 1 (VERDADEIRO) se as geometrias atenderem às condições especificadas pela cadeia de matrizes do padrão DE-9IM; do contrário, 0 (FALSO) retornará. Para obter informações sobre as matrizes de padrão DE-9IM, consulte “Funções do predicado” na página 141.

Sintaxe

db2gse.ST_Relate(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry, patternMatrix String)

Tipo de retorno

Inteiro

Exemplos

Uma matriz de padrão DE-9IM é um dispositivo para comparação de geometrias. Há vários tipos de matrizes. Por exemplo, a matriz do padrão *é igual* informará se as duas geometrias são iguais.

Neste exemplo, uma matriz de padrão igual, mostrada em Tabela 56, é lida da esquerda para a direita e de cima para baixo numa cadeia (“T*F**FFF*”).

Tabela 56. Matriz de padrão igual

		b		
		Interior	**Limite**	**Exterior**
	Interior	T	*	F
a	**Limite**	*	*	F
	Exterior	F	F	*

Em seguida, a tabela RELATE_TEST é criada com a seguinte instrução CREATE TABLE.

```
CREATE TABLE RELATE_TEST (rid integer, g1 db2gse.ST_Geometry,
g2 db2gse.ST_Geometry, g3 db2gse.ST_Geometry);
```

As seguintes instruções INSERT inserem uma subclasse na tabela RELATE_TEST.

```
INSERT INTO RELATE_TEST VALUES(
       1,
       db2gse.ST_PointFromText('point (10.02 20.01)',
            db2gse.coordref()..srid(0),
       db2gse.ST_PointFromText('point (10.02 20.01)',
            db2gse.coordref()..srid(0),
        db2gse.ST_PointFromText('point (30.01 20.01)',
db2gse.coordref()..srid(0)
)
```

A seguinte instrução SELECT e o conjunto de resultados correspondente relaciona o nome da subclasse armazenado na coluna GEOTYPE à dimensão desse geotipo.

```
SELECT rid, relate(g1,g2) equals, relate(g1,g3) not_equals
FROM relate_test

RID     equals    not_equals
-------- -----------   -----------------
1         1              0

  1 record(s) selected.
```

ST_SRID

ST_SRID toma um objeto de geometria e retorna sua identidade do sistema de referência espacial.

Sintaxe

db2gse.ST_SRID(g1 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

Durante a instalação do DB2 Spatial Extender, a tabela SPATIAL_REFERENCES é criada. Quando uma geometria é criada, o SRID dessa geometria é inserido na tabela SPATIAL_REFERENCES. A função ST_SRID retorna o valor dessa entrada.

Por exemplo, um tipo de geometria é usado na instrução CREATE TABLE:

```
CREATE TABLE SRID_TEST(g1 db2gse.ST_Geometry)
```

Na seguinte instrução INSERT, é inserido uma geometria do ponto localizado na coordenada 10.01,50.76 na coluna de geometria G1. Quando a geometria do ponto foi criada pela função ST_PointFromText, ela foi atribuída ao valor srid 1.

```
INSERT INTO SRID_TEST
VALUES (db2gse.ST_PointFromText('point(10.01 50.76)',

db2gse.coordref()..srid(0)))
```

A função ST_SRID retorna a identidade do sistema de referência espacial da geometria que acaba de ser fornecida, conforme ilustrado pela seguinte instrução SELECT e pelo conjunto de resultados correspondente.

```
SELECT db2gse.ST_SRID(g1) FROM SRID_TEST

g1
--------------
1
```

ST_StartPoint

ST_StartPoint toma uma cadeia de linha e retorna um ponto que indica o primeiro ponto das cadeias de linhas.

Sintaxe

db2gse.ST_StartPoint(c db2gse.ST_Curve)

Tipo de retorno

db2gse.ST_Point

Exemplos

A seguinte instrução CREATE TABLE cria a tabela STARTPOINT_TEST. STARTPOINT_TEST tem duas colunas: a coluna de inteiros GID, que identifica exclusivamente as linhas da tabela e a coluna da cadeia de linhas LN1.

```
CREATE TABLE STARTPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

As seguintes instruções INSERT inserem as cadeias de linhas na coluna LN1. A primeira cadeia de linha não possui coordenadas Z ou medidas, ao passo que a segunda possui ambas.

```
INSERT INTO STARTPOINT_TEST VALUES(1,
db2gse.ST_LineFromText('linestring (10.02 20.01,23.73
21.92,30.10 40.23)', db2gse.coordref()..srid(0)))

INSERT INTO STARTPOINT_TEST VALUES(2,
db2gse.ST_LineFromText('linestring  zm (10.02 20.01 5.0 7.0,
    23.73 21.92 6.5 7.1,30.10 40.23 6.9 7.2)',

db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostram a função ST_StartPoint extraindo o primeiro ponto de cada cadeia de linha. A função ST_AsText converte o ponto ao seu formato de texto. O primeiro ponto na lista não possui uma coordenada Z ou uma medida, enquanto o segundo ponto tem ambos, porque a cadeia de linhas de origem tinha.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.ST_StartPoint (ln1)) as varchar(60))
"Startpoint"
FROM STARTPOINT_TEST

GID         Startpoint
----------- ------------------------------------------------------------
          1 POINT ( 10.02000000 20.01000000)
          2 POINT ZM ( 10.02000000 20.01000000 5.00000000 7.00000000)

  2 record(s) selected.
```

ST_SymmetricDiff

O ST_SymmetricDiff toma dois objetos de geometria e retorna um objeto de geometria que é a diferença simétrica dos objetos de origem.

Sintaxe

db2gse.ST_SymmetricDiff(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

O supervisor do município deve determinar a área sensível e o raio de 8 km de áreas de risco que não têm interseção.

A seguinte instrução CREATE TABLE cria a tabela SENSITIVE_AREAS, que contém várias colunas que descrevem as instituições ameaçadas. A tabela SENSITIVE_AREAS também contém a coluna ZONE, que armazena a geometria do polígono da instituição.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

A seguinte instrução CREATE TABLE cria a tabela HAZARDOUS_SITES, que armazena a identidade dos locais nas colunas SITE_ID e NAME, ao passo que a localização geográfica real de cada local está armazenada na coluna de ponto LOCATION.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  point);
```

A função ST_Buffer gera um buffer de 8 km ao redor das áreas do lixo tóxico. A função ST_SymmetricDiff gera polígonos a partir da intersecção dos polígonos de locais com lixo tóxico em buffer e das áreas sensíveis. A função ST_Area retorna a área de polígono da interseção, de cada área de risco.

```
SELECT sa.name, hs.name,
       db2gse.ST_Area(db2gse.ST_SymmetricDiff (db2gse.ST_Buffer(hs.location,
(5 * 5280)),sa.zone))
FROM HAZARDOUS_SITES hs, SENSITIVE_AREAS sa
```

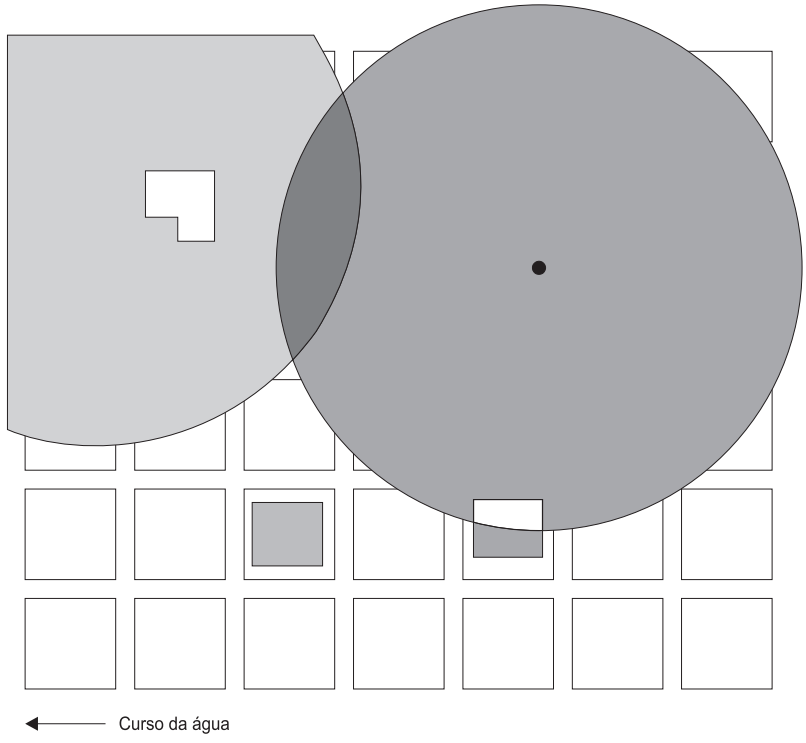


Figura 38. Utilizando o ST_SymmetricDiff para determinar as áreas de lixo tóxico que não contêm áreas sensíveis (construções desabitadas)

Na Figura 38, a diferença simétrica das áreas de lixo tóxico e das área sensíveis resulta na subtração das áreas de interseção.

ST_Touches

ST_Touches retorna 1 (VERDADEIRO) se nenhum dos pontos comuns a ambas as geometrias fizerem interseção com os interiores das duas; do contrário, retornará 0 (FALSO). Pelo menos uma geometria deve ser uma cadeia de linhas, polígono, cadeia de linhas múltiplas ou multipolígono.

Sintaxe

db2gse.ST_Touches(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

O técnico GIS precisa fornecer uma lista de todas as linhas de sewer cujas extremidades façam interseção com outra linha de sewer.

A seguinte instrução CREATE TABLE cria a tabela SEWERLINES, a qual possui três colunas. A primeira coluna, SEWER_ID, identifica exclusivamente cada linha do sewer. A segunda coluna, CLASS, do tipo inteiro identifica o tipo de linha de sewer, que geralmente está associada à capacidade da linha. A terceira coluna, SEWER, do tipo cadeia de linhas armazena a geometria da linha sewer.

```
CREATE TABLE SEWERLINES (sewer_id integer, class integer, sewer
    db2gse.ST_LineString);
```

A seguinte instrução SELECT retorna uma lista ordenada de SEWER_IDS que se encontram.

```
SELECT s1.sewer_id, s2.sewer_id
FROM sewerlines s1, sewerlines s2
WHERE db2gse.ST_Touches (s1.sewer, s2.sewer) = 1,
ORDER BY 1,2;
```

ST_Transform

O ST_Transform atribui uma geometria a um sistema de referência espacial diferente do sistema de referência espacial ao qual a geometria está atribuída atualmente.

Sintaxe

db2gse.ST_Transform(g db2gse.ST_Geometry, cr db2gse.coordref)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

A seguinte instrução CREATE TABLE cria a tabela TRANSFORM_TEST, que tem duas colunas de cadeia de linhas, L1 e L2.

```
CREATE TABLE TRANSFORM_TEST (tid integer, l1 db2gse.ST_LineString,
l2 db2gse.ST_LineString)
```

A seguinte instrução INSERT insere uma cadeia de linhas em l1 com SRID de 102.

```
INSERT INTO TRANSFORM_TEST VALUES (1, db2gse.ST_LineFromText('linestring
    (10.01 40.43, 92.32 29.89)',
    db2gse.coordref()..srid(102)),NULL)
```

A função ST_Transform converte a cadeia de linhas de L1 da referência de coordenada atribuída para SRID 102 à referência de coordenadas atribuída para SRID 105. A seguinte instrução UPDATE armazena a cadeia de linhas transformada na coluna l2.

```
UPDATE TRANSFORM_TEST SET l2 = db2gse.ST_Transform(l1,
    db2gse.coordref()..srid(105))
```

ST_Union

ST_Union soma dois objetos de geometria e retorna um objeto de geometria que é a união dos objetos de origem.

Sintaxe

db2gse.ST_Union(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

db2gse.ST_Geometry

Exemplos

A seguinte instrução CREATE TABLE cria a tabela SENSITIVE_AREAS, que contém várias colunas que descrevem as instituições ameaçadas. A tabela SENSITIVE_AREAS também contém a coluna ZONE, que armazena a geometria do polígono da instituição.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

A seguinte instrução CREATE TABLE cria a tabela HAZARDOUS_SITES, que armazena a identidade dos locais nas colunas SITE_ID e NAME. A localização geográfica real de cada área está armazenada na coluna de ponto LOCATION.

```
CREATE TABLE HAZARDOUS_SITES (site_id integer, name varchar(128),
                              location  db2gse.ST_Point);
```

A seguinte instrução SELECT usa a função ST_Buffer para gerar um buffer de 8 km ao redor das áreas de lixo tóxico. A função ST_Union gera polígonos a partir da união dos polígonos de locais de lixo tóxico em buffer e das áreas sensíveis. A função ST_Area retorna a união da área do polígono.

```
SELECT sa.name, hs.name,
       db2gse.ST_Area(db2gse.ST_Union(db2gse.ST_Buffer(hs.location,
(5 * 5280)),sa.zone))
FROM HAZARDOUS_SITES hs, SENSITIVE_AREAS sa;
```

ST_Within

ST_Within toma dois objetos de geometria e retorna 1 (VERDADEIRO) se o primeiro objeto estiver completamente dentro do segundo; do contrário, retornará 0 (FALSO).

Sintaxe

db2gse.ST_Within(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo de retorno

Inteiro

Exemplos

No exemplo abaixo são criadas duas tabelas. A primeira, BUILDINGFOOTPRINTS, contém as bases de construção da cidade. A segunda tabela, LOTS, contém os lotes da cidade. O engenheiro da cidade deseja verificar se todas as bases de construção estão inteiramente dentro de seus lotes.

Nas duas tabelas, o tipo de dados de multipolígono armazena a geometria das bases e lotes de construção. O projetista selecionou os multipolígonos para os dois recursos, porque os lotes podem ser desmembrados por recursos naturais, como um rio e que as bases de construção podem ser geralmente constituídas de vários edifícios.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);


CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

Utilizando a seguinte instrução SELECT, o engenheiro da cidade primeiro seleciona as construções que não estão completamente dentro de um lote.

```
SELECT building_id
  FROM BUILDINGFOOTPRINTS, LOTS
 WHERE db2gse.ST_Within(footprint,lot) = 0;
```

Embora a primeira consulta forneça uma lista de todos os BUILDINGS_IDs que têm bases foram de um polígono do lote, ela não determinará se o restante possui a id de lote atribuído a eles. Esta segunda consulta executa uma verificação de integridade dos dados na coluna LOT_ID da tabela BUILDINGFOOTPRINTS.

```
SELECT bf.building_id "building id",
       bf.lot_id "buildings lot_id",
       LOTS.lot_id "LOTS lot_id"
FROM BUILDINGFOOTPRINTS bf, LOTS
 WHERE db2gse.ST_Within(footprint,lot) = 1 AND
       LOTS.lot_id <> bf.lot_id;
```

ST_WKBToSQL

ST_WKBToSQL constrói um valor de ST_Geometry dada a sua representação do modo binário reconhecida. Um valor 0 de SRID é usado automaticamente.

Sintaxe

db2gse.ST_WKBToSQL(WKBGeometry Blob(1M))

Tipo de retorno

db2gse.ST_Geometry

Exemplos

A instrução CREATE TABLE a seguir cria a tabela LOTS, que possui duas colunas: a coluna LOT_ID, que identifica exclusivamente cada lote, e a coluna de multipolígono LOT, que contém a geometria de cada lote.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

O seguinte fragmento de código C contém funções ODBC incorporadas às funções SQL do DB2 Spatial Extender que inserem dados na tabela LOTS.

A função ST_WKBToSQL converte representações de WKB em geometria do DB2 Spatial Extender. A instrução INSERT inteira é copiada em uma cadeia wkb_sql char. A instrução INSERT contém marcadores do parâmetro para a ceitar os dados do ID_do LOT e os dados do LOTE dinamicamente.

```
/* Criar a instrução insert do SQL para preencher a id do lote e o
  multipolígono do lote. Os pontos de interrogação são marcadores
  do parâmetro que  indicam a id do_lote e os valores do lote
  que serão recuperados no   tempo de execução.*/

strcpy (wkb_sql,"insert into lots (lot_id, lot)
  values(?, db2gse.ST_WKBToSQL(cast(? as blob(1m))))");

/* Alocar memória para o tratamento da instrução SQL e associar o
   tratamento da instrução ao tratamento da conexão. */

rc = SQLAllocStmt (handle, &hstmt);

/* Preparar as instruções SQL para execução. */

rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Fazer o bind do valor inteiro da chave ao primeiro parâmetro.*/

pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Fazer o bind do shape ao segundo parâmetro. */

pcbvalue2 = blob_len;
```

```
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Executar a instrução insert . */

rc = SQLExecute (hstmt);
```

ST_WKTToSQL

ST_WKTToSQL constrói um valor de ST_Geometry dada a sua representação textual reconhecida. Um valor 0 de SRID é usado automaticamente.

Sintaxe

db2gse.ST_WKTToSQL(geometryTaggedText Varchar(4000))

Tipo de retorno

db2gse.ST_Geometry

Exemplos

A seguinte tabela CREATE TABLE cria a tabela GEOMETRY_TEST, que contém duas colunas: a coluna GID do tipo inteiro, que identifica exclusivamente cada linha, e a coluna G1, que armazena a geometria.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

As seguintes instruções INSERT inserem os dados nas colunas GID e G1 da tabela GEOMETRY_TEST. A função ST_WKTToSQL converte a representação de texto de cada geometria na subclasse de instâncias do DB2 Spatial Extender correspondente.

```
INSERT INTO GEOMETRY_TEST VALUES(
1, db2gse.ST_WKTToSQL ('point (10.02 20.01)')
)

INSERT INTO GEOMETRY_TEST VALUES(
2, db2gse.ST_WKTToSQL('linestring (10.01 20.01, 10.01 30.01, 10.01 40.01)')
)

INSERT INTO GEOMETRY_TEST VALUES(
3, db2gse.ST_WKTToSQL('polygon ((10.02 20.01, 11.92 35.64, 25.02 34.15,
19.15 33.94, 10.02 20.01))')
)

INSERT INTO GEOMETRY_TEST VALUES(
4, db2gse.ST_WKTToSQL('multipoint (10.02 20.01,10.32 23.98,11.92 25.64)')
)

INSERT INTO GEOMETRY_TEST VALUES(
5, db2gse.ST_WKTToSQL('multilinestring ((10.02 20.01,      10.32 23.98,
11.92 25.64),(9.55 23.75,15.36 30.11))')
)

INSERT INTO GEOMETRY_TEST VALUES(
6, db2gse.ST_WKTToSQL('multipolygon (((10.02 20.01, 11.92 35.64,
25.02 34.15, 19.15 33.94,10.02 20.01)),
((51.71 21.73, 73.36 27.04, 71.52 32.87, 52.43 31.90, 51.71 21.73)))')
)
```

ST_X

ST_X toma um ponto e retorna sua coordenada X.

Sintaxe

ST_X(p ST_Point)

Tipo de retorno

Duplo

Exemplos

A seguinte tabela CREATE TABLE cria a tabela X_TEST, que tem duas colunas: a coluna GID, que identifica exclusivamente a linha, e a coluna de ponto PT1.

```
CREATE TABLE X_TEST (gid integer, pt1 db2gse.ST_Point)
```

A seguinte instrução INSERT insere duas linhas. Uma é um ponto sem uma coordenada Z ou uma medida. A outra coluna possui uma coordenada Z e uma medida.

```
INSERT INTO X_TEST VALUES(1,
db2gse.ST_PointFromText('point (10.02 20.01)', db2gse.coordref()..srid(0)))

INSERT INTO X_TEST VALUES(2,
db2gse.ST_PointFromText('point  zm (10.02 20.01 5.0 7.0)',

db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondentes relacionam a coluna GID e a coordenada Double X dos pontos.

```
SELECT gid, db2gse.ST_X(pt1) "The X coordinate" FROM X_TEST

GID         The X coordinate
----------- ------------------------
          1   +1.00200000000000E+001
          2   +1.00200000000000E+001

  2 record(s) selected.
```

ST_Y

ST_Y toma um ponto e retorna sua coordenada Y.

Sintaxe

db2gse.ST_Y(p db2gse.ST_Point)

Tipo de retorno

Duplo

Exemplos

A seguinte tabela CREATE TABLE cria a tabela Y_TEST, que tem duas colunas: a coluna GID, que identifica exclusivamente a linha, e a coluna de ponto PT1.

```
CREATE TABLE Y_TEST (gid integer, pt1 db2gse.ST_Point)
```

A instrução INSERT insere duas linhas. Uma é um ponto sem uma coordenada Z ou uma medida. A outra coluna possui uma coordenada Z e uma medida.

```
INSERT INTO Y_TEST VALUES(1,
db2gse.ST_PointFromText('point (10.02 20.01)', db2gse.coordref()..srid(0)))

INSERT INTO Y_TEST VALUES(2,
db2gse.ST_PointFromText('point  zm (10.02 20.01 5.0 7.0)',

db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente relacionam a coluna GID e a coordenada Double Y dos pontos.

```
SELECT gid, db2gse.ST_Y(pt1) "The Y coordinate" FROM Y_TEST

GID         The Y coordinate
----------- ------------------------
          1   +2.00100000000000E+001
          2   +2.00100000000000E+001

  2 record(s) selected.
```

Z

Z toma um ponto e retorna sua coordenada Z.

Sintaxe

Z(p db2gse.ST_Point)

Tipo de retorno

Duplo

Exemplos

A seguinte tabela CREATE TABLE cria a tabela Z_TEST, que tem duas colunas: a coluna GID, que identifica exclusivamente a linha, e a coluna de ponto PT1.

```
CREATE TABLE Z_TEST (gid integer, pt1 db2gse.ST_Point)
```

A seguinte instrução INSERT insere duas linhas. Uma é um ponto sem uma coordenada Z ou uma medida. A outra coluna possui uma coordenada Z e uma medida.

```
INSERT INTO Z_TEST VALUES(1,
db2gse.ST_PointFromText('point (10.02 20.01)', db2gse.coordref()..srid(0)))

INSERT INTO Z_TEST VALUES(2,
db2gse.ST_PointFromText('point  zm (10.02 20.01 5.0 7.0)',

db2gse.coordref()..srid(0)))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente relacionam a coluna GID e a coordenada Double Z dos pontos. A primeira linha é NULA porque o ponto não tem uma coordenada Z.

```
SELECT gid, db2gse.Z(pt1) "The Z coordinate" FROM Z_TEST

GID         The Z coordinate
----------- ------------------------
          1                        -
          2   +5.00000000000000E+000

  2 record(s) selected.
```

Capítulo 15. Sistemas de coordenadas

O capítulo fornece informações de referência sobre o sistema de referência espacial (SRS) e sobre os valores de coordenadas utilizados para interpretar dados espaciais.

Visão geral dos sistemas de coordenadas

A representação de texto reconhecida dos sistemas de referência espacial fornece uma representação textual padrão para informações sobre o sistema de referência. As definições da representação reconhecida são modeladas após o modelo de dados do sistema de coordenadas do POSC/EPSG.

Um sistema de referência espacial é uma (latitude-longitude) geográfica, um (X,Y) projetado ou um sistema de coordenadas (X,Y,Z) geocêntrico. O sistema de coordenadas é composto de vários objetos. Cada objeto tem uma palavra-chave em maiúsculas (por exemplo, DATUM ou UNIT) seguido dos parâmetros de definição, delimitados por vírgulas, do objeto entre colchetes. Alguns objetos são compostos de outros objetos, portanto, o resultado é uma estrutura aninhada.

Nota: As implementações estão livres para substituir os colchetes padrão ( ) por chaves [ ] e estar preparadas para ler os dois formatos.

A definição EBNF (Extended Backus Naur Form) para a representação de cadeia de um sistema de coordenadas que usa colchetes é como segue (consulte a nota acima sobre o uso de colchetes):

```
<sistema de coordenadas> = <cs projetado> | <cs geográfico> | <cs geocêntrico>
<cs projetado> = PROJCS["<nome>", <cs geográfico>, <projeção>, {<parâmetro>,}*
                 <unidade linear>]
<projeção> = PROJECTION["<nome>"]
<parâmetro> = PARAMETER["<nome>", <valor>]
<valor> = <número>
```

Um sistema de coordenadas do arquivo será identificado pela palavra-chave PROJCS se os dados estiverem em coordenadas projetadas (por GEOGCS se em coordenadas geográficas ou por GEOCCS se em coordenadas geocêntricas). A palavra-chave PROJCS é seguida de todas as partes″ que definem o sistema de coordenadas projetadas. A primeira parte de qualquer objeto é sempre o nome. Vários objetos seguem o nome do sistema de coordenadas projetadas: o sistema de coordenadas geográficas, a projeção de mapas, um ou mais parâmetros e a unidade linear de medida. Todos os sistemas de coordenadas projetadas são baseados num sistema de coordenadas geográficas, portanto, esta sessão descreve primeiro as partes específicas de um sistema de coordenadas projetadas. Por exemplo, a zona UTM 10N nos dados NAD83 está definida:

```
PROJCS["NAD_1983_UTM_Zone_10N",
<cs geográfico>,
PROJECTION["Transverse_Mercator"],
PARAMETER["Falso_Leste".500000,0],
PARAMETER["Falso_Norte".0,0],
PARAMETER["Central_Meridiano",-123,0],
PARAMETER["Escala_Fator".0,9996],
PARAMETER["Latitude_de_Origem".0,0],
UNIT["Metro".1,0]]
```

O nome e vários objetos definem o objeto do sistema de coordenadas geográficas em turnos: o dado, o meridiano principal e a unidade angular de medida.

```
<cs geográfico> = GEOGCS["<nome>", <dado>, <meridiano principal>,
<unidade angular>] <dado> = DATUM["<nome>", <esferóide>]
<esferóide> = SPHEROID["<nome>", <eixo semi-principal>, <condensação inversa>]
<eixo semi-principal> = <número>
    (o eixo semiprincipal é medido em metros e deve ser > 0.)
<condensação inversa> = <número>
<meridiano principal> = PRIMEM["<nome>", <longitude>]
<longitude> = <número>
```

A cadeia do sistema de coordenadas geográficas para a zona UTM 10 em NAD83:

```
GEOGCS["GCS_Norte_americano_1983",
DATUM["D_Norte_americano_1983",
SPHEROID["GRS_1980".6378137.298,257222101]],
PRIMEM["Greenwich",0],
UNIT["Grau".0,0174532925199433]]
```

O objeto UNIT pode representar unidade angular ou linear de medidas:

```
<unidade angular> = <unidade>
<unidade linear> = <unidade>
<unidade> = UNIT["<nome>", <fator de conversão>]
<fator de conversão> = <número>
```

O fator de conversão especifica o número de metros (para uma unidade linear) ou o número de radianos (para uma unidade angular) por unidade e deve ser maior que zero.

Assim, a representação completa da cadeia da Zona UTM 10N é a seguinte:

```
PROJCS["NAD_1983_UTM_Zone_10N",
GEOGCS["GCS_Norte_americano_1983",
DATUM[ "D_Norte_americano_1983",SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Grau",0.0174532925199433]],
PROJECTION["Transverse_Mercator"],PARAMETER["Falso_Leste",500000.0],
PARAMETER["Falso_Norte",0.0],PARAMETER["Meridiano_Central",-123.0],
PARAMETER["Escala_Fator",0.9996],PARAMETER["Latitude_de_Origem",0.0],
UNIT["Metro",1.0]]
```

Um sistema de coordenadas geométricas é semelhante a um sistema de coordenadas geográficas:

```
<cs geocêntrico> = GEOCCS["<nome>", <dado>, <meridiano principal>,
<unidade linear>]
```

Unidades lineares suportadas

Tabela 57. Unidades lineares suportadas

Unidade	Fator de conversão
Metro	1,0
Pé (Internacional)	0,3048
Pé americano	12 / 39,37
Pé americano modificado	12,0004584 / 39,37
Pé de Clarke	12 / 39,370432
Pé indiano	12 / 39,370141
Ligação	7.92 / 39.370432
Ligação (Benoit)	7.92 / 39.370113
Ligação (Sears)	7.92 / 39.370147
Cadeia (Benoit)	792 / 39.370113
Cadeia (Sears)	792 / 39.370147
Jarda (Indian)	36 / 39.370141
Jarda (Sears)	36 / 39.370147

Tabela 57. Unidades lineares suportadas (continuação)

Unidade	Fator de conversão
Braça	1.8288
Milha náutica	1852.0

Unidades angulares suportadas

Tabela 58. Unidades angulares suportadas

Unidade	Fator de conversão
Radiano	1.0
Grau Decimal	p / 180
Minuto Decimal	(p / 180) / 60
Segundo Decimal	(p / 180) / 36000
Gon	p / 200
Grade	p / 200

Esferóides suportados

Tabela 59. Esferóides suportados

Nome	Eixo semi-principal	Condensação inversa
Airy	6377563.396	299.3249646
Airy modificado	6377340.189	299.3249646
Australiano	6378160	298.25
Bessel	6377397.155	299.1528128
Bessel modificado	6377492.018	299.1528128
Bessel (Namíbia)	6377483.865	299.1528128
Clarke 1866	6378206.4	294.9786982
Clarke 1866 (Michigan)	6378693.704	294.978684677
Clarke 1880	6378249.145	293.465
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)	6378249.2	293.46598

Tabela 59. Esferóides suportados (continuação)

Nome	Eixo semi-principal	Condensação inversa
Everest 1830	6377276.345	300.8017
Everest 1975	6377301.243	300.8017
Everest (Sarawak e Sabah)	6377298.556	300.8017
Everest modificado 1948	6377304.063	300.8017
Fischer 1960	6378166	298.3
Fischer 1968	6378150	298.3
Fischer modificado (1960)	6378155	298.3
GEM10C	6378137	298.257222101
GRS 1980	6378137	298.257222101
Hayford 1909	6378388	297.0
Helmert 1906	6378200	298.3
Hough	6378270	297.0
Internacional 1909	6378388	297.0
Internacional 1924	6378388	297.0
Novo Internacional 1967	6378157.5	298.2496
Krasovsky	6378245	298.3
Mercury 1960	6378166	298.3
Mercury modificado 1968	6378150	298.3
NWL9D	6378145	298.25
OSU_86F	6378136.2	298.25722
OSU_91A	6378136.3	298.25722
Plessis 1817	6376523	308.64
América do Sul 1969	6378160	298.25
Ásia do Sul	6378155	298.3
Esfera (raio = 1.0)	1	0
Esfera (raio = 6371000 m)	6371000	0
Esfera (raio =6370997 m)	6370997	0
Struve 1860	6378297	294.73
Walbeck	6376896	302.78
War Office	6378300.583	296
WGS 1960	6378165	298.3
WGS 1966	6378145	298.25

Tabela 59. Esferóides suportados (continuação)

Nome	Eixo semi-principal	Condensação inversa
WGS 1972	6378135	298.26
WGS 1984	6378137	298.257223563

Dados geodéticos suportados

Tabela 60. Dados geodéticos suportados

Adindan	Lisboa
Afgooye	Loma Quintana
Agadez	Lome
Dados geodéticos australianos 1966	Luzon 1911
Dados geodéticos australianos 1984	Mahe 1971
Ain el Abd 1970	Makassar
Amersfoort	Malongo 1987
Aratu	Manoca
Arc 1950	Massawa
Arc 1960	Merchich
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Bogota	Nord de Guerre
Bukit Rimpah	NGO 1948
Camacupa	Nord Sahara 1959
Campo Inchauspe	NSWC 9Z-2
Cape	Nouvelle Triangulation Francaise
Carthage	New Zealand Geodetic Datum 1949

Tabela 60. Dados geodéticos suportados (continuação)

Chua	OS (SN) 1980
Conakry 1905	OSGB 1936
Corrego Alegre	OSGB 1970 (SN)
Cote d'Ivoire	Padang 1884
Datum 73	Palestina 1923
Deir ez Zor	Pointe Noire
Deutsche Hauptdreiecksnetz	Provisional South American Datum 1956
Douala	Pulkovo 1942
European Datum 1950	Qatar
European Datum 1987	Qatar 1948
Egito 1907	Qornoq
European Reference System 1989	RT38
Fahud	South American Datum 1969
Gandajika 1970	Sapper Hill 1943
Garoua	Schwarzeck
Geocentric Datum of Australia 1994	Segora
Guiana Francesa	Serindung
Herat North	Estocolmo 1938
Hito XVIII 1963	Sudão
Hu Tzu	Shan Tananarive 1925
Hungarian Datum 1972	Timbalai 1948
Indian 1954	TM65
Indian 1975	TM75
Indonesian Datum 1974	Tóquio
Jamaica 1875	Trinidad 1903
Jamaica 1969	Trucial Coast 1948
Kalianpur	Voirol 1875
Kandawala	Voirol Unifie 1960
Kertau	WGS 1972
Kuwait Oil Company	WGS 1972 Transit Broadcast Ephemeris
La Canoa	WGS 1984
Lake	Yacare
Leigon	Yoff

Tabela 60. Dados geodéticos suportados (continuação)

Libéria 1964	Zanderij

Meridianos principais suportados

Tabela 61. Meridianos principais suportados

Localização	Coordenadas
Greenwich	0° 0' 0"
Bern	7° 26' 22.5" E
Bogotá	74° 4' 51.3" W
Bruxelas	4° 22' 4.71" E
Ferro	17° 40' 0" W
Jakarta	106° 48' 27.79" E
Lisboa	9° 7' 54.862" W
Madri	3° 41' 16.58" W
Paris	2° 20' 14.025" E
Roma	12° 27' 8.4" E
Estocolmo	18° 3' 29" E

Projeções do mapa suportadas

Tabela 62. Projeções do mapa suportadas

Projeções cilíndricas	Projeções pseudocilíndricas
Behrmann	Craster parabolic
Cassini	Eckert I
Área igual cilíndrica	Eckert II
Equiretangular	Eckert III
Estereográfico de Gall	Eckert IV
Gauss-Kruger	Eckert V
Mercator	Eckert VI
Miller cilindrico	McBryde-Thomas flat polar quartic
Oblíquo	Mercator (Hotine) Mollweide
Plate-Carée	Robinson
Times	Sinusoidal (Sansom-Flamsteed)

Tabela 62. Projeções do mapa suportadas (continuação)

Projeções cilíndricas	Projeções pseudocilíndricas
Transverse Mercator	Winkel I

Projeções cônicas

Tabela 63. Projeções cônicas

Área igual cônica de Albers	Chamberlin trimétrico
Cônico conformal oblíquo bipolar	Dois pontos equidistantes
Bonne	Área igual Hammer-Aitoff
Cônico equidistante	Van der Grinten I
Cônico conformal de Lambert	Diversos
Policônico	Alaska série E
Cônico simples	Grade Alaska (Esterográfico modificado por Snyder)

Projeções azimutais ou em circuito impresso

- Azimutal equidistante
- Perspectiva geral do lado direito vertical
- Gnomônico
- Área igual de Lambert Azimutal
- Ortográfico
- Polar-Esterográfico
- Esterográfico

Parâmetros de projeção do mapa

Tabela 64. Parâmetros de projeção do mapa

Parâmetro	Descrição
meridiano_central	A linha da longitude escolhida como a origem das coordenadas x.
fator_de_escala	Utilizado geralmente para reduzir a quantidade de distorção na projeção de um mapa.
padrão_paralelo_1	Uma linha de latitude que geralmente não apresenta distorção. Usada também para ″latitude de escala verdadeira.″

Tabela 64. Parâmetros de projeção do mapa (continuação)

Parâmetro	Descrição
padrão_paralelo_2	Uma linha de latitude que geralmente não apresenta distorção.
longitude_do_centro	A longitude que define o ponto central da projeção do mapa.
latitude_do_centro	A latitude que define o ponto central da projeção do mapa.
latitude_da_origem	A latitude escolhida como a origem das coordenadas y.
leste_falso	Incluído em coordenadas x. Usado para fornecer valores positivos.
norte_falso	Incluído em coordenadas y. Usado para fornecer valores positivos.
azimute	O ângulo leste do norte que define a linha central de uma projeção oblíqua.
longitude_de_ponto_1	A longitude do primeiro ponto necessário para uma projeção de mapa.
latitude_de_ponto_1	A latitude do primeiro ponto necessário para uma projeção de mapa.
longitude_de_ponto_2	A longitude do segundo ponto necessário para uma projeção de mapa.
latitude_de_ponto_2	A latitude do segundo ponto necessário para uma projeção de mapa.
longitude_de_ponto_3	A longitude do terceiro ponto necessário para uma projeção de mapa.
latitude_de_ponto_3	A latitude do terceiro ponto necessário para uma projeção de mapa.
número de_landsat	O número de um satélite Landsat.
número_do caminho	O número de caminho orbital de um determinado satélite.
peso_de_ponto de_perspectiva	O peso acima da terra do ponto de perspectiva da projeção do mapa.
fipszone	Número de zona do Sistema de Coordenadas Planas do Estado.
zona	Número de zona UTM.

Capítulo 16. Formatos de arquivos para dados espaciais

Este capítulo documenta as representações reconhecidas do DB2 Spatial Extender. As representações estão descritas como *reconhecidas* pois foram fornecidas por ESRI e não são específicas de DB2 Spatial Extender. Três tipos de valores espaciais são importantes para entender a importação e exportação de dados espaciais:

- A representação de texto reconhecida OGC (Open GIS Consortium)
- As representações (WKB) do modo binário reconhecidas de OGC
- As representações de shape ESRI

As representações de texto reconhecidas de OGC

O DB2 Spatial Extender possui várias funções que geram geometrias a partir de descrições de textos:

ST_GeomFromText
Cria uma geometria a partir da representação de texto de qualquer tipo de geometria.

ST_PointFromText
Cria um ponto a partir de uma representação de texto de ponto.

ST_LineFromText
Cria uma cadeia de linhas a partir de uma representação de texto da cadeia de linhas.

ST_PolyFromText
Cria um polígono a partir de uma representação do texto do polígono.

ST_MPointFromText
Cria um multiponto a partir de representações de textos multipontos.

ST_MLineFromText
Cria uma cadeia de linhas múltiplas a partir de uma representação de texto de cadeia de linhas múltiplas.

ST_MPolyFromText
Cria um multipolígono a partir de uma representação de texto de multipolígono.

A representação de texto é uma cadeia de formato de texto ASCII que permite que a geometria seja trocada no formato de texto ASCII. Você pode usar estas funções num programa (3GL ou 4GL) de linguagem de terceira ou quarta

geração mas elas não exigem as definições de estruturas de programas especiais. A função ST_AsText converte uma geometria existente numa representação de texto.

Cada tipo de geometria possui uma representação de texto reconhecida, que pode ser usada para construir novas instâncias do tipo e converter instâncias existentes na forma textual para exibição alfanumérica.

A representação de texto reconhecida de uma geometria é definida da seguinte forma: a notação {}* denota 0 ou mais repetições dos tokens dentro das chaves; as chaves não aparecem na lista de tokens de saída.

```
<Texto Marcado de Geometria> :=
| <Texto Marcado de Ponto>
| <Texto Marcado de Cadeia de Linha>
| <Texto Marcado de Polígono>
| <Texto Marcado de MultiPonto>
| <Texto Marcado de Cadeia de Linhas Múltiplas>
| <Texto Marcado de MultiPolígono>

<Texto Marcado de Ponto> :=
POINT <Texto de Ponto>

<Texto Marcado de Cadeia de Linha> :=
LINESTRING <Texto de Cadeia de Linha>

<Texto Marcado de Polígono> :=
POLYGON <Texto de Polígono>

<Texto Marcado de MultiPonto> :=
MULTIPOINT <Texto de Multiponto>

<Texto Marcado de Cadeia de Linhas Múltiplas> :=
MULTILINESTRING <Texto de Cadeia de Linhas Múltiplas>

<Texto Marcado de MultiPolígono> :=
MULTIPOLYGON <Texto de MultiPolígono>

<Texto de Ponto> := EMPTY
|    <Ponto>
| Z  <PontoZ>
| M  <PontoM>
| ZM <PontoZM>

<Ponto> :=  <x>  <y>
<x> := literal de precisão dupla
<y> := literal de precisão dupla
<PontoZ> :=  <x>  <y>  <z>
<x> := literal de precisão dupla
<y> := literal de precisão dupla
<z> := literal de precisão dupla
<PontoM> :=  <x>  <y>  <m>
<x> := literal de precisão dupla
<y> := literal de precisão dupla
```

```
<m> := literal de precisão dupla
<PontoZM> :=  <x>  <y>  <z>  <m>
<x> := literal de precisão dupla
<y> := literal de precisão dupla
<z> := literal de precisão dupla
<m> := literal de precisão dupla

<Texto de Cadeia de Linha> := EMPTY
|    ( <Texto de Ponto >   {,  <Texto de Ponto > }*  )
| Z  ( <Texto de Ponto Z >  {,  <Texto de Ponto Z> }*  )
| M  ( <Texto de Ponto M >  {,  <Texto de Ponto M> }*  )
| ZM ( <Texto de Ponto ZM > {,  <Texto de Ponto ZM> }*  )

<Texto de Polígono> := EMPTY
| ( <Texto de Cadeia de Linha > {,< Texto de Cadeia de Linha > }*  )

<Texto de Multiponto> := EMPTY
| ( <Texto de Ponto >   {,  <Texto de Ponto > }*  )

<Texto de Cadeia de Linhas Múltiplas> := EMPTY
| ( <Texto de Cadeia de Linha > {,< Texto de Cadeia de Linha > }*  )

<Texto de MultiPolígono> := EMPTY
| ( < Texto de Polígono > {,  < Texto de Polígono > }*  )
```

A sintaxe da função básica é:

```
função (<descrição do texto>,<SRID>)
```

O SRID, o identificador de referência espacial e a chave primária da tabela SPATIAL_REFERENCES, identifica o sistema de referência espacial que está armazenado na tabela SPATIAL_REFERENCES. Antes de uma geometria ser inserida numa coluna espacial, seu SRID deve corresponder ao SRID da coluna espacial.

A descrição do texto é composta de três componentes básicos que são colocados entre aspas, por exemplo:

```
<'tipo de geometria'> ['tipo de coordenada'] [''lista de coordenada']
```

em que:

tipo de geometria
: É um dos seguintes: ponto, cadeia de linha, polígono, multiponto, cadeia de linhas múltiplas ou multipolígono.

tipo de coordenada
: Especifica se a geometria terá coordenadas Z ou medidas. Deixe este argumento em branco se nenhum deles existir na geometria. Caso contrário, defina o tipo de coordenada em Z para geometrias que contém coordenadas Z, M para geometrias com medidas e ZM para geometrias que possuam ambas.

lista de coordenadas

Define os vértices da geometria. As listas de coordenadas são delimitadas por vírgulas e colocadas entre parênteses. As geometrias com vários componentes exigem conjuntos de parênteses abrindo e fechando cada parte do componente. Se a geometria estiver vazia, a palavra-chave EMPTY substituirá a coordenada.

A Tabela 65 mostra uma lista completa de exemplos de todas as representações de texto possíveis.

Tabela 65. Tipos de geometria e suas representações de texto

Tipo de geometria	Descrição do texto	Comentário
ponto	ponto vazio	ponto vazio
ponto	ponto z vazio	ponto vazio com coordenada z
ponto	ponto m vazio	ponto vazio com medida
ponto	ponto zm vazio	ponto vazio com coordenada z e medida
ponto	ponto (10.05 10.28)	ponto
ponto	ponto z (10.05 10.28 2.51)	ponto com coordenada z
ponto	ponto m (10.05 10.28 4.72)	ponto com medida
ponto	ponto zm (10.05 10.28 2.51 4.72)	ponto com coordenada z e medida
cadeia de linha	cadeia de linha vazia	cadeia de linha vazia
cadeia de linha	cadeia de linha z vazia	cadeia de linha vazia com coordenadas z
cadeia de linha	cadeia de linha m vazia	cadeia de linha vazia com medidas
cadeia de linha	cadeia de linha zm vazia	cadeia de linha vazia com coordenadas z e medidas
cadeia de linha	cadeia de linha(10.05 10.28 , 20.95 20.89)	cadeia de linha
cadeia de linha	cadeia de linha z (10.05 10.28 3.09, 20.95 31.98 4.72, 21.98 29.80 3.51)	cadeia de linha com coordenadas z
cadeia de linha	cadeia de linha m (10.05 10.28 5.84, 20.95 31.98 9.01, 21.98 29.80 12.84)	cadeia de linha com medidas
cadeia de linha	cadeia de linha zm ( )	cadeia de linha com coordenadas z e medidas
polígono	polígono vazio	polígono vazio

Tabela 65. Tipos de geometria e suas representações de texto (continuação)

Tipo de geometria	Descrição do texto	Comentário
polígono	polígono z vazio	polígono vazio com coordenadas z
polígono	polígono m vazio	polígono vazio com medidas
polígono	polígono zm vazio	polígono vazio com coordenadas z e medidas
polígono	polígono ((10 10, 10 20, 20 20, 20 15, 10 10))	polígono
polígono	polígono z (( ))	polígono com coordenadas z
polígono	polígono m (( ))	polígono com medidas
polígono	polígono zm (( ))	polígono com coordenadas z e medidas
multiponto	multiponto vazio	multiponto vazio
multiponto	multiponto z vazio	multiponto vazio com coordenadas z
multiponto	multiponto m vazio	multiponto vazio com medidas
multiponto	multiponto zm vazio	multiponto vazio com coordenadas z com medidas
multiponto	multiponto vazio	multiponto vazio
multiponto	multiponto (10 10, 20 20)	multiponto com dois pontos
multiponto	multiponto z (10 10 2, 20 20 3)	multiponto com coordenadas z
multiponto	multiponto m (10 10 4, 20 20 5)	multiponto com medidas
multiponto	multiponto zm (10 10 2 4, 20 20 3 5)	multiponto com coordenadas z e medidas
cadeia de linhas múltiplas	cadeia de linhas múltiplas vazia	cadeia de linhas múltiplas vazia
cadeia de linhas múltiplas	cadeia de linhas múltiplas z vazia	cadeia de linhas múltiplas vazia com coordenadas z
cadeia de linhas múltiplas	cadeia de linhas múltiplas m vazia	cadeia de linhas múltiplas vazia com medidas

Tabela 65. Tipos de geometria e suas representações de texto (continuação)

Tipo de geometria	Descrição do texto	Comentário
cadeia de linhas múltiplas	cadeia de linhas múltiplas zm vazia	cadeia de linhas múltiplas vazia com coordenadas z e medidas
cadeia de linhas múltiplas	cadeia de linhas múltiplas (( ))	cadeia de linhas múltiplas
cadeia de linhas múltiplas	cadeia de linhas múltiplas z (( ))	cadeia de linhas múltiplas com coordenadas z
cadeia de linhas múltiplas	cadeia de linhas múltiplas m (( ))	cadeia de linhas múltiplas com medidas
cadeia de linhas múltiplas	cadeia de linhas múltiplas zm (( ))	cadeia de linhas múltiplas com coordenadas z e medidas
multipolígono	multipolígono vazio	multipolígono vazio
multipolígono	multipolígono z vazio	multipolígono vazio com coordenadas z
multipolígono	multipolígono m vazio	multipolígono vazio com medidas
multipolígono	multipolígono z	multipolígono vazio com coordenadas z e medidas
multipolígono	multipolígono ((( )))	multipolígono
multipolígono	multipolígono z ((( )))	multipolígono com coordenadas z
multipolígono	multipolígono m (((10 10 2, 10 20 3, 20 20 4, 20 15 5, 10 10 2), (50 40 7, 50 50 3, 60 50 4, 60 40 5, 50 40 7)))	multipolígono com medidas
multipolígono	multipolígono zm ((( )))	multipolígono com coordenadas z e medidas

As representações (WKB) do modo binário reconhecidas

DB2 Spatial Extender possui várias funções que geram geometrias a partir de representações do modo binário:

ST_GeomFromWKB
Cria uma geometria a partir de uma representação WKB de qualquer tipo de geometria.

ST_PointFromWKB
Cria um ponto a partir de uma representação de WKB.

ST_LineFromWKB
Cria uma cadeia de linhas a partir de uma representação de WKB.

ST_PolyFromWKB
Cria um polígono a partir de uma representação de WKB do polígono.

ST_MPointFromWKB
Cria um multiponto a partir de uma representação de WKB.

ST_MLineFromWKB
Cria uma cadeia de linhas múltiplas a partir de uma representação de WKB de cadeia de linhas múltiplas.

ST_MPolyFromWKB
Cria um multipolígono a partir de uma representação de WKB de multipolígono.

A representação do modo binário reconhecido é um fluxo contíguo de bytes. Ela permite que a geometria sejam trocada entre um cliente ODBC e um banco de dados SQL no modo binário. Como estas funções de geometria exigem a definição das estruturas da linguagem de programação para mapear a representação do modo binário, elas destinam-se ao uso dentro de um programa da linguagem de terceira geração (3GL). Elas não se adaptam ao ambiente da linguagem de quarta geração (4GL). A função ST_AsBinary converte um valor de geometria existente numa representação do modo binário reconhecida.

A representação do modo binário reconhecido para geometria é obtida pela serialização de uma instância da geometria como uma seqüência de tipos numéricos. Estes tipos são retirados do conjunto (inteiro não-assinado, duplo) e, cada tipo numérico é serializado como uma seqüência de bytes. Os tipos são serializados através de uma das representações padrão, do modo binário, bem-definidas para tipos numéricos (NDR, XDR). Uma tag de um byte que precede os bytes serializados descreve a codificação binária específica (NDR ou XDR) usada para um fluxo de bytes de geometria. A única diferença entre as duas codificações da geometria é uma da ordem de bytes: A codificação XDR é Big Endian; a codificação NDR é Little Endian.

Definições do tipo numérico

Um *inteiro não-assinado* é um tipo de dados (4 bytes) de 32 bits que codifica um inteiro não-negativo no intervalo [0, 4294967295].

Um *duplo* é um tipo de dados de precisão dupla de (8 bytes) e 64 bits que codifica um número de precisão dupla através do formato de precisão dupla 754 do IEEE.

Estas definições são comuns para XDR e NDR.

XDR (Big Endian) codificação de tipos numéricos

A representação XDR de um inteiro não-assinado é Big Endian (byte mais significativo primeiro).

A representação XDR de um duplo é Big Endian (bit de sinal é o primeiro byte).

NDR (Little Endian) codificação de tipos numéricos

A representação NDR de um inteiro não-assinado é Little Endian (byte menos significativo primeiro).

A representação NDR de um duplo é Little Endian (bit de sinal é o último byte).

Conversão entre NDR e XDR

A conversão entre os tipos de dados NDR e XDR para inteiros não-assinados e duplos é uma operação simples. Ela envolve a inversão da ordem de bytes dentro de cada inteiro não-assinado ou duplo no fluxo de bytes.

Descrição dos fluxos de bytes WKBGeometry

Esta seção descreve a representação do modo binário reconhecida para geometria. O bloco de construção básica é o fluxo de bytes para um ponto, que consiste em dois duplos. Os fluxos de bytes para outras geometrias são construídos através de fluxos de bytes para geometrias que já estejam definidas.

```
// Definições do tipo básico
// byte : 1 byte
// uint32 : inteiro não-assinado de 32 bits (4 bytes)
// double : número de precisão dupla (8 bytes)

// Construindo Blocos : Ponto, Anel Linear

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
  Ponto    point;
};
WKBLineString {
  byte     byteOrder;
  uint32   wkbType;                                          // 2
  uint32   numPoints;
  Point    points[numPoints];
}

WKBPolygon    {
  byte                byteOrder;
  uint32            wkbType;                                 // 3
  uint32            numRings;
  LinearRing        rings[numRings];
}
WKBMultiPoint    {
  byte                byteOrder;
  uint32            wkbType;                                 // 4
  uint32            num_wkbPoints;
  WKBPoint            WKBPoints[num_wkbPoints];
}
WKBMultiLineString    {
  byte              byteOrder;
  uint32            wkbType;                                 // 5
  uint32            num_wkbLineStrings;
  WKBLineString     WKBLineStrings[num_wkbLineStrings];
}

wkbMultiPolygon {
  byte              byteOrder;
  uint32            wkbType;                                 // 6
  uint32            num_wkbPolygons;
  WKBPolygon        wkbPolygons[num_wkbPolygons];
}

WKBGeometry  {
  union {
    WKBPoint                point;
    WKBLineString           linestring;
    WKBPolygon              polygon;
    WKBMultiPoint           mpoint;
    WKBMultiLineString      mlinestring;
    WKBMultiPolygon         mpolygon;
  }
};
```

A figura a seguir mostra uma representação NDR.

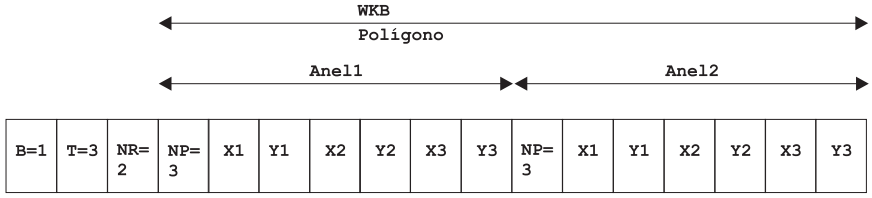


Figura 39. Representação no formato NDR. (B=1) do tipo polígono (T=3) com 2 lineares (NR=2), cada anel tendo 3 pontos (NP=3).

Assertivas para a representação WKB

A representação do modo binário reconhecida para geometria foi projetada para representar as instâncias dos tipos de geometria descritos no Modelo do Objeto de Geometria e Especificação do Abstrato da OpenGIS.

Estas assertiva implicam o seguinte para anéis, polígonos e multipolígonos:

Anéis lineares
Os anéis são simples e fechados, significando que os anéis lineares não podem fazer interseção entre si.

Polígonos
Dois anéis lineares no limite de um polígono não podem se cruzarem. Os anéis lineares no limite de um polígono podem fazer interseção no máximo em um único ponto, mas somente como tangente.

Multipolígonos
Os interiores de dois polígonos que são elementos de um multipolígono não podem fazer interseção. Os limites de qualquer polígono que seja elemento de um multipolígono podem tocar somente um número finito de pontos.

As representações de shape ESRI

O DB2 Spatial Extender possui várias funções que geram geometrias a partir de representações de shape ESRI. Além da representação bidimensional suportada pela representação do modo binário de GIS aberto, representação de shape ESRI também suporta as coordenadas Z opcionais e medidas. As seguintes funções geram geometria a partir de um shape ESRI:

ST_GeometryFromShape
Cria uma geometria a partir de uma representação de shape de qualquer tipo de geometria.

ST_PointFromShape
Cria um ponto a partir de uma representação de shape.

ST_LineFromShape
Cria uma cadeia de linhas a partir de uma representação de shape.

ST_PolyFromShape
Cria um polígono a partir de uma representação de shape do polígono.

ST_MPointFromShape
Cria um multiponto a partir de uma representação do shape.

ST_MLineFromShape
Cria uma cadeia de linhas múltiplas a partir de uma representação de shape da cadeia de linhas múltiplas.

ST_MPolyFromShape
Cria um multipolígono a partir de uma representação de shape de multipolígono.

A sintaxe geral destas funções é a mesma. O primeiro argumento é a representação de shape fornecida como um tipo de dados do objeto grande do modo binário (BLOB). O segundo argumento é o identificador de referência espacial para atribuir a geometria. A função GeometryFromShape tem a seguinte sintaxe:

```
db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)
```

Como estas funções de shape exigem a definição das estruturas da linguagem de programação C para mapear a representação binária, elas destinam-se ao uso dentro de um programa 3GL e não se adequam a um ambiente 4GL. A função AsShape converte o valor da função em uma representação de shape ESRI.

Um tipo de shape 0 indica um shape nulo, sem dados geométricos para o shape.

Valor	Tipo de Shape
0	Shape Nulo
1	Ponto
3*	PoliLinha
5	Polígono
8	MultiPonto
11	PontoZ
13	PoliLinhaZ
15	PolígonoZ
18	MultiPontoZ

Valor	Tipo de Shape
21	PontoM
23	PoliLinhaM
25	PolígonoM
28	MultiPontoM

Nota: * Os tipos de shapes não especificados acima (2, 4, 6 etc.) são reservados para uso futuro.

Tipos de shapes no espaço XY

Ponto

Um ponto consiste num par de coordenadas de precisão dupla na ordem X, Y.

Tabela 66. Conteúdo do fluxo de bytes do ponto

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	1	Inteiro	1	Little
Byte 4	X	X	Duplo	1	Little
Byte 12	Y	Y	Duplo	1	Little

MultiPonto

Um MultiPonto consiste numa coleção de pontos. O quadro delimitador é armazenado na ordem Xmin, Ymin, Xmax, Ymax.

Tabela 67. Conteúdo do fluxo de bytes MultiPonto

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	8	Inteiro	1	Little
Byte 4	Caixa	Caixa	Duplo	4	Little
Byte 36	NumPoints	NumPoints	Inteiro	1	Little
Byte 40	Pontos	Pontos	Ponto	NumPoints	Little

Polilinha

Uma Polilinha é um conjunto ordenado de vértices que consiste em uma ou mais partes. Uma parte é uma seqüência conectada de dois ou mais pontos. Os pontos podem ou não estar conectados entre si. As partes podem ou não fazer interseção entre si.

Como esta especificação não proíbe pontos consecutivos com coordenadas idênticas, os leitores do arquivo de shape deverão manipular tais casos. Por outro lado, as partes degeneradas, de comprimento zero que poderão resultar não são permitidas.

Os campos para uma Polilinha são:

Quadro
O quadro delimitador da Polilinha armazenada na ordem Xmin, Ymin, Xmax, Ymax.

NumParts
O número de partes na Polilinha.

NumPoints
O número total de pontos para todas as partes.

Partes Uma matriz do comprimento NumParts. Cada Polilinha armazena o índice de seu primeiro ponto na matriz de pontos. Os índices matriciais são relativos a 0.

Pontos
Uma matriz do comprimento NumPoints. Os pontos de cada parte na Polilinha são armazenados de ponto a ponto. Os pontos para a parte 2 seguem os pontos da parte 1 etc. A matriz das partes mantém o índice matricial do ponto inicial para cada parte. Não existe delimitador na matriz de pontos entre as partes.

Tabela 68. Conteúdo do fluxo de bytes Polilinha

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	3	Inteiro	1	Little
Byte 4	Caixa	Caixa	Duplo	4	Little
Byte 36	NumParts	NumParts	Inteiro	1	Little
Byte 40	NumPoints	NumPoints	Inteiro	1	Little
Byte 44	Partes	Partes	Inteiro	NumParts	Little
Byte X	Pontos	Pontos	Ponto	NumPoints	Little

Nota: X = 44 + 4 * NumParts.

Polígono
Um polígono consiste num ou mais anéis. Um anel é uma seqüência conectada de quatro ou mais pontos que forma um loop fechado, sem auto-interseção. Um polígono pode conter vários anéis externos. A ordem dos vértices ou orientação de um anel indica qual lado do anel é o interior do polígono. A área à direita do observador que caminha pelo anel na ordem do

vértice é a área dentro do polígono. Os vértices de anéis que definem os orifícios nos polígonos estão no sentido anti-horário. Os vértices de um único polígono de anel, portanto, estão sempre no sentido horário. Os anéis de um polígono são denominados partes.

Como esta especificação não proíbe pontos consecutivos com coordenadas idênticas, os leitores do arquivo de shape deverão manipular tais casos. Por outro lado, as partes degeneradas, de comprimento zero ou de área zero que poderão resultar não são permitidas.

Os campos para um polígono são:

Quadro
O quadro delimitador do polígono armazenado na ordem Xmin, Ymin, Xmax, Ymax.

NumParts
O número de anéis no polígono.

NumPoints
O número total de pontos para todos os anéis.

Partes Uma matriz do comprimento NumParts. Para cada anel, armazena o índice de seu primeiro ponto na matriz de pontos. Os índices matriciais são relativos a 0.

Pontos
Uma matriz do comprimento NumPoints. Os pontos de cada anel no polígono são armazenados de ponto a ponto. Os pontos do Anel 2 seguem os pontos do Anel 1 etc. A matriz das partes mantém o índice matricial do ponto inicial para cada anel. Não existe delimitador na matriz de pontos entre anéis.

Avisos importantes sobres os shapes do polígono:

- Os anéis estão fechados (o primeiro e o último vértice de um anel DEVEM ser o mesmo).
- A ordem dos anéis na matriz de pontos não é significativa.
- Os polígonos armazenados num arquivo de shape devem estar limpos. Um polígono limpo é um que:
  - Não possui auto-interseções. Isto significa que um segmento pertencente a um anel não pode fazer interseção com um segmento que pertença a outro anel. Os anéis de um polígono podem se tocar nos vértices mas não nos segmentos. Os segmento colineares são considerados interseção.
  - Tem o interior de um polígono no lado ″correto″ da linha que o define. A área à direita do observador que passa pelo anel na ordem do vértice é o interior do polígono. Os vértices de um

único polígono de anel, portanto, estão sempre no sentido horário. Os anéis que definem orifícios nestes polígonos têm um sentido anti-horário.

Polígonos ″sujos″ ocorrem quando os anéis que definem os orifícios no polígono também estão no sentido horário, o que causa a sobreposição dos interiores.

Um Exemplo de Instância de Polígono:

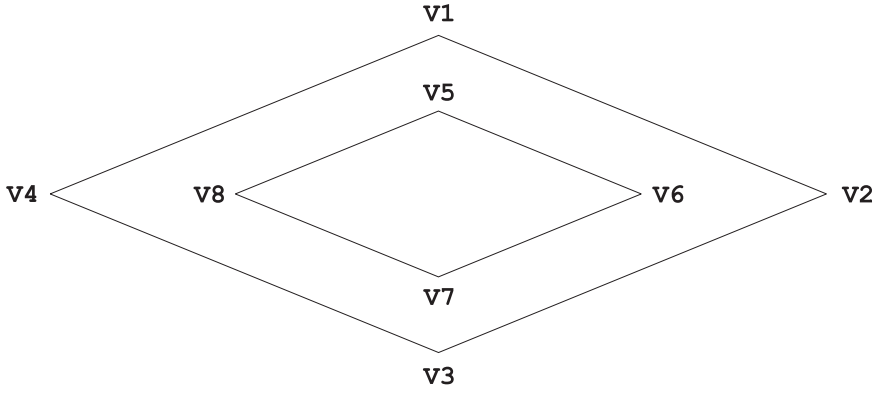


Figura 40. Um polígono com um orifício e oito vértices

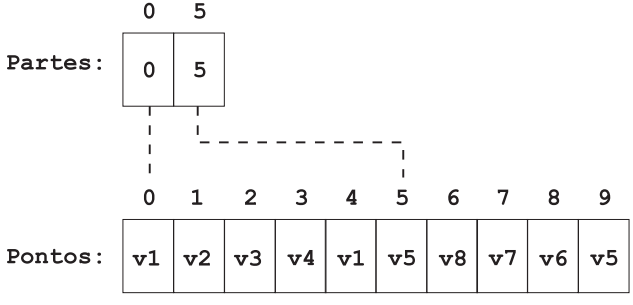


Figura 41. Conteúdo do fluxo de bytes do polígono. NumParts igual a 2 e NumPoints igual a 10. Observe que a ordem dos pontos para o polígono (orifício) do donut é inversa.

Tabela 69. Conteúdo do fluxo de dados do polígono

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	5	Inteiro	1	Little
Byte 4	Caixa	Caixa	Duplo	4	Little
Byte 36	NumParts	NumParts	Inteiro	1	Little

Tabela 69. Conteúdo do fluxo de dados do polígono (continuação)

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 40	NumPoints	NumPoints	Inteiro	1	Little
Byte 44	Partes	Partes	Inteiro	NumParts	Little
Byte X	Pontos	Pontos	Ponto	NumPoints	Little

Nota: X = 44 + 4 * NumParts.

Tipos de shapes medidos no espaço XY

PointM

Um PointM consiste num par de coordenadas de precisão dupla na ordem X, Y, mais uma medida M.

Tabela 70. Conteúdo do fluxo de bytes do pontoM

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	21	Inteiro	1	Little
Byte 4	X	X	Duplo	1	Little
Byte 12	Y	Y	Duplo	1	Little
Byte 20	M	M	Duplo	1	Little

MultiPontoM

Os campos para um MultiPontoM são:

Quadro
O quadro delimitador do MultipontoM armazenado na ordem Xmin, Ymin, Xmax, Ymax.

NumPoints
O número de pontos.

Pontos
Uma matriz de Pontos do comprimento NumPoints.

NumMs
O número de Medidas que seguem. NumMs pode ter somente dois valores zero se nenhuma Medida seguir este campo; ou igual a NumPoints se Medidas estiverem presentes.

Faixa M
As medidas mínimas e máximas para o MultipontoM armazenado na ordem Mmin, Mmax.

Matriz M
Uma matriz de Medidas do comprimento NumPoints.

Tabela 71. Conteúdo do fluxo de bytes do MultipontoM

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	28	Inteiro	1	Little
Byte 4	Caixa	Caixa	Duplo	4	Little
Byte 36	NumPoints	NumPoints	Inteiro	1	Little
Byte 40	Pontos	Pontos	Ponto	NumPoints	Little
Byte X	NumMs	NumMs	Inteiro	1	Little
Byte X+4*	Mmin	Mmin	Duplo	1	Little
Byte X+12*	Mmax	Mmax	Duplo	1	Little
Byte X+20*	Marray	Marray	Duplo	NumPoints	Little

Notas:

1. X = 40 + (16 * NumPoints)
2. * opcional

PolilinhaM

Um arquivo de shape PolilinhaM consiste em uma ou mais partes. Uma parte é uma seqüência conectada de dois ou mais pontos. As partes podem ou não estar conectadas entre si. As partes podem ou não fazer interseção entre si.

Os campos para uma PolilinhaM são:

Quadro
O quadro delimitador da PolilinhaM armazenado na ordem Xmin, Ymin, Xmax, Ymax.

NumParts
O número de partes na PolilinhaM.

NumPoints
O número total de pontos para todas as partes.

Partes Uma matriz do comprimento NumParts. Para cada parte, armazena o índice de seu primeiro ponto na matriz de pontos. Os índices matriciais são relativos a 0.

Pontos
Uma matriz do comprimento NumPoints. Os pontos de cada parte na PolilinhaM são armazenados de ponto a ponto. Os pontos para a parte 2 seguem os pontos da parte 1 etc. A matriz das partes mantém

o índice matricial do ponto inicial para cada parte. Não existe delimitador na matriz de pontos entre as partes.

NumMs

O número de Medidas que seguem. NumMs pode ter somente dois valores zero se nenhuma Medida seguir este campo; ou igual a NumPoints se Medidas estiverem presentes.

Faixa M

As medidas mínimas e máximas para a PolilinhaM armazenada na ordem Mmin, Mmax.

Matriz M

Uma matriz do comprimento NumPoints. As medidas de cada parte na PolilinhaM são armazenadas de ponto a ponto. As medidas para a parte 2 seguem as medidas da parte 1 etc. A matriz das partes mantém o índice matricial do ponto inicial para cada parte. Não existe delimitador na matriz de medida entre as partes.

Tabela 72. Conteúdo do fluxo de bytes da PolilinhaM

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	13	Inteiro	1	Little
Byte 4	Caixa	Caixa	Duplo	4	Little
Byte 36	NumParts	NumParts	Inteiro	1	Little
Byte 40	NumPoints	NumPoints	Inteiro	1	Little
Byte 44	Partes	Partes	Inteiro	NumParts	Little
Byte X	Pontos	Pontos	Ponto	NumPoints	Little
Byte Y	NumMs	NumMs	Inteiro	1	Little
Byte Y+4*	Mmin	Mmin	Duplo	1	Little
Byte Y+12*	Mmax	Mmax	Duplo	1	Little
Byte Y+20*	Marray	Marray	Duplo	NumPoints	Little

Notas:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * opcional

PolígonoM

Um PolígonoM consiste num número de anéis. Um anel é um loop fechado, sem auto-interseção. Observe que interseções são calculadas no espaço XY, *não* no espaço XYM. Um PolígonoM pode conter vários anéis externos. Os anéis de um PolígonoM são denominados partes.

Os campos para um PolígonoM são:

Quadro
O quadro delimitador do PolígonoM armazenado na ordem Xmin, Ymin, Xmax, Ymax.

NumParts
O número de anéis no PolígonoM.

NumPoints
O número total de pontos para todos os anéis.

Partes Uma matriz do comprimento NumParts. Para cada anel, armazena o índice de seu primeiro ponto na matriz de pontos. Os índices matriciais são relativos a 0.

Pontos
Uma matriz do comprimento NumPoints. Os pontos de cada anel no PolígonoM são armazenados de ponto a ponto. Os pontos do Anel 2 seguem os pontos do Anel 1 etc. A matriz das partes mantém o índice matricial do ponto inicial para cada anel. Não existe delimitador na matriz de pontos entre anéis.

NumMs
O número de Medidas que seguem. NumMs pode ter somente dois valores zero se nenhuma Medida seguir este campo, ou igual a NumPoints se Medidas estiverem presentes.

Faixa M
As medidas mínimas e máximas para o PolígonoM armazenado na ordem Mmin, Mmax.

Matriz M
Uma matriz do comprimento NumPoints. As medidas de cada anel no PolígonoM são armazenados de ponto a ponto. As medidas do Anel 2 seguem os pontos do Anel 1 etc. A matriz das partes mantém o índice matricial da medida inicial para cada anel. Não existe delimitador na matriz de medida entre os anéis.

Avisos importantes sobre os shapes do PolígonoM:

- Os anéis estão fechados (o primeiro e o último vértice de um anel devem ser o mesmo).
- A ordem dos anéis na matriz de pontos não é significativa.

Tabela 73. Conteúdo do fluxo de bytes do PolígonoM

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	15	Inteiro	1	Little
Byte 4	Caixa	Caixa	Duplo	4	Little
Byte 36	NumParts	NumParts	Inteiro	1	Little

Tabela 73. Conteúdo do fluxo de bytes do PolígonoM (continuação)

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 40	NumPoints	NumPoints	Inteiro	1	Little
Byte 44	Partes	Partes	Inteiro	NumParts	Little
Byte X	Pontos	Pontos	Ponto	NumPoints	Little
Byte Y	NumMs	NumMs	Inteiro	1	Little
Byte Y+4*	Mmin	Mmin	Duplo	1	Little
Byte Y+12*	Mmax	Mmax	Duplo	1	Little
Byte Y+20*	Marray	Marray	Duplo	NumPoints	Little

Notas:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * opcional

Tipos de shapes no espaço XYZ

PontoZ

Um PontoZ consiste em coordenadas triplas e duplas na ordem de X, Y, Z mais uma medida.

Tabela 74. Conteúdo do fluxo de bytes do PontoZ

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	11	Inteiro	1	Little
Byte 4	X	X	Duplo	1	Little
Byte 12	Y	Y	Duplo	1	Little
Byte 20	Z	Z	Duplo	1	Little
Byte 28	Medida	M	Duplo	1	Little

MultipontoZ

Um MultipontoZ representa um conjunto de PontosZ, da seguinte forma:

- O quadro delimitador é armazenado na ordem Xmin, Ymin, Xmax, Ymax.
- O intervalo delimitador de Z é armazenado na ordem Zmin, Zmax. O intervalo delimitador de M é armazenado na ordem Mmin, Mmax.

Tabela 75. Conteúdo do fluxo de bytes do MultipontoZ

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	18	Inteiro	1	Little

Tabela 75. Conteúdo do fluxo de bytes do MultipontoZ (continuação)

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 4	Caixa	Caixa	Duplo	4	Little
Byte 36	NumPoints	NumPoints	Inteiro	1	Little
Byte 40	Pontos	Pontos	Ponto	NumPoints	Little
Byte X	Zmin	Zmin	Duplo	1	Little
Byte X+8	Zmax	Zmax	Duplo	1	Little
Byte X+16	Zarray	Zarray	Duplo	NumPoints	Little
Byte Y	NumMs	NumMs	Inteiro	1	Little
Byte Y+4*	Mmin	Mmin	Duplo	1	Little
Byte Y+12*	Mmax	Mmax	Duplo	1	Little
Byte Y+20*	Marray	Marray	Duplo	NumPoints	Little

Notas:

1. X = 40 + (16 * NumPoints); Y = X + 16 + (8 * NumPoints)
2. * opcional

PolilinhaZ

Uma PolilinhaZ consiste em uma ou mais partes. Uma parte é uma seqüência conectada de dois ou mais pontos. As partes podem ou não estar conectadas entre si. As partes podem ou não fazer interseção entre si.

Os campos para uma PolilinhaZ são:

Quadro
O quadro delimitador da PolilinhaZ armazenado na ordem Xmin, Ymin, Xmax, Ymax.

NumParts
O número de partes na PolilinhaZ.

NumPoints
O número total de pontos para todas as partes.

Partes Uma matriz do comprimento NumParts. Para cada parte, armazena o índice de seu primeiro ponto na matriz de pontos. Os índices matriciais são relativos a 0.

Pontos
Uma matriz do comprimento NumPoints. Os pontos de cada parte na PolilinhaZ são armazenados de ponto a ponto. Os pontos para a parte 2 seguem os pontos da parte 1 etc. A matriz das partes mantém o índice matricial do ponto inicial para cada parte. Não existe delimitador na matriz de pontos entre as partes.

Faixa Z

Os valores mínimos e máximo de Z para a PolilinhaZ armazenada na ordem Zmin, Zmax.

Matriz Z

Uma matriz do comprimento NumPoints. Os valores de Z para cada parte na PolilinhaZ são armazenados de ponto a ponto. Os valores de Z para a parte 2 seguem os valores de Z para a parte 1 e assim por diante. A matriz das partes mantém o índice matricial do ponto inicial para cada parte. Não existe delimitador na matriz Z entre as partes.

NumMs

O número de Medidas que seguem. NumMs pode ter somente dois valores zero se nenhuma Medida seguir este campo; ou igual a NumPoints se Medidas estiverem presentes.

Faixa M

As medidas mínimas e máximas para a PolilinhaZ armazenada na ordem Mmin, Mmax.

Matriz M

Uma matriz do comprimento NumPoints. As medidas para cada parte na PolilinhaZ são armazenados de ponto a ponto. As medidas para a parte 2 seguem as medidas da parte 1 etc. A matriz das partes mantém o índice matricial da medida inicial para cada parte. Não existe delimitador na matriz de medida entre as partes.

Tabela 76. Conteúdo do fluxo de bytes da PolilinhaZ

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	13	Inteiro	1	Little
Byte 4	Caixa	Caixa	Duplo	4	Little
Byte 36	NumParts	NumParts	Inteiro	1	Little
Byte 40	NumPoints	NumPoints	Inteiro	1	Little
Byte 44	Partes	Partes	Inteiro	NumParts	Little
Byte X	Pontos	Pontos	Ponto	NumPoints	Little
Byte Y	Zmin	Zmin	Duplo	1	Little
Byte Y+8	Zmax	Zmax	Duplo	1	Little
Byte Y+16	Zarray	Zarray	Duplo	NumPoints	Little
Byte Z	NumMs	NumMs	Inteiro	1	Little
Byte Z+4*	Mmin	Mmin	Duplo	1	Little
Byte Z+12*	Mmax	Mmax	Duplo	1	Little
Byte Z+20*	Marray	Marray	Duplo	NumPoints	Little

Notas:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints), Z = Y + 16 + (8 * NumPoints)
2. * opcional

PolígonoZ

Um PolígonoZ consiste num número de anéis. Um anel é um loop fechado, sem auto-interseção. Um PolígonoZ pode conter vários anéis externos. Os anéis de um PolígonoZ são denominados partes.

Os campos para um PolígonoZ são:

Quadro
O quadro delimitador do PolígonoM armazenado na ordem Xmin, Ymin, Xmax, Ymax.

NumParts
O número de anéis no PolígonoZ.

NumPoints
O número total de pontos para todos os anéis.

Partes Uma matriz do comprimento NumParts. Para cada anel, armazena o índice de seu primeiro ponto na matriz de pontos. Os índices matriciais são relativos a 0.

Pontos
Uma matriz do comprimento NumPoints. Os pontos de cada anel no PolígonoZ são armazenados de ponto a ponto. Os pontos do Anel 2 seguem os pontos do Anel 1 etc. A matriz das partes mantém o índice matricial do ponto inicial para cada anel. Não existe delimitador na matriz de pontos entre anéis.

Faixa Z
Os valores mínimos e máximo de Z para o arco armazenado na ordem Zmin, Zmax.

Matriz Z
Uma matriz do comprimento NumPoints. Os valores de Z para cada anel no PolígonoZ são armazenados de ponto a ponto. Os valores de Z para o Anel 2 seguem os valores de Z para o Anel 1 e assim por diante. A matriz das partes mantém o índice matricial do valor Z inicial para cada anel. Não existe delimitador na matriz do valor Z entre os anéis.

NumMs
O número de Medidas que seguem. NumMs pode ter somente dois valores zero se nenhuma Medida seguir este campo; ou igual a NumPoints se Medidas estiverem presentes.

Faixa M

As medidas mínimas e máximas para o PolígonoZ armazenado na ordem Mmin, Mmax.

Matriz M

Uma matriz do comprimento NumPoints. As medidas para cada anel no PolígonoZ são armazenados de ponto a ponto. As medidas do Anel 2 seguem os pontos do Anel 1 etc. A matriz das partes mantém o índice matricial da medida inicial para cada anel. Não existe delimitador na matriz de medida entre os anéis.

Avisos importantes sobre os shapes do PolígonoZ:

- Os anéis estão fechados (o primeiro e o último vértice de um anel DEVEM ser o mesmo).
- A ordem dos anéis na matriz de pontos não é significativa.

Tabela 77. Conteúdo do fluxo de bytes do PolígonoZ

Posição	Campo	Valor	Tipo	Número	Solicitar
Byte 0	Tipo de Shape	15	Inteiro	1	Little
Byte 4	Caixa	Caixa	Duplo	4	Little
Byte 36	NumParts	NumParts	Inteiro	1	Little
Byte 40	NumPoints	NumPoints	Inteiro	1	Little
Byte 44	Partes	Partes	Inteiro	NumParts	Little
Byte X	Pontos	Pontos	Ponto	NumPoints	Little
Byte Y	Zmin	Zmin	Duplo	1	Little
Byte Y+8	Zmax	Zmax	Duplo	1	Little
Byte Y+16	Zarray	Zarray	Duplo	NumPoints	Little
Byte Z	NumMs	NumMs	Inteiro	1	Little
Byte Z+4*	Mmin	Mmin	Duplo	1	Little
Byte Z+12*	Mmax	Mmax	Duplo	1	Little
Byte Z+20*	Marray	Marray	Duplo	NumPoints	Little

Parte 3. Apêndices

Avisos

A IBM pode não oferecer os produtos, serviços ou recursos discutidos neste documento em todos os países. Consulte seu representante IBM local para informações sobre os produtos e serviços atualmente disponíveis em sua área. Qualquer referência a um produto, programa ou serviço IBM não pretende declarar ou subentender que apenas este produto IBM, programa ou serviço possa ser utilizado. Qualquer produto, programa ou serviço funcionalmente equivalente que não infrinja qualquer direito de propriedade intelectual da IBM pode ser utilizado em substituição. De qualquer modo, é de responsabilidade do usuário avaliar e verificar a operação de qualquer produto, programa ou serviço não-IBM.

A IBM pode possuir patentes ou solicitações de patentes pendentes relativas a assuntos descritos nesta publicação. O fornecimento desta publicação não lhe garante direito algum sobre tais patentes. Você pode enviar consultas sobre patentes, por escrito, para:

Gerente de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138 / 146
Botafogo
22290-240, Rio de Janeiro - RJ
Brasil

O seguinte parágrafo não se aplica ao Reino Unido ou a qualquer outro país onde tais permissões são inconsistentes com as lei locais: A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO "COMO ESTÁ" SEM QUALQUER TIPO DE GARANTIA, EXPRESSA OU IMPLÍCITA, INCLUINDO, PORÉM NÃO LIMITADA A, GARANTIAS IMPLÍCITAS DE NÃO-VIOLAÇÃO, MERCANTABILIDADE OU AJUSTES PARA UM PROPÓSITO PARTICULAR. Alguns estados não permitem negar garantias expressas ou implícitas em certas transações, portanto estas instruções talvez não se aplicam a você.

Estas informações podem possuir imprecisões técnicas ou erros tipográficos. Alterações são feitas periodicamente às informações aqui contidas; estas informações serão incorporadas em novas edições da publicação. A IBM pode fazer melhorias e/ou alterações no(s) produto(s) e/ou programa(s) descritos nesta publicação em qualquer tempo sem qualquer aviso prévio.

Quaisquer referências nestas informações a sites Web não-IBM são fornecidas por conveniência apenas e de nenhuma maneira serve como uma aprovação

daqueles sites Web. Os materiais daqueles sites Web não são parte dos materiais para este produto IBM e a utilização dos mesmos é por seu próprio risco.

A IBM pode utilizar ou distribuir qualquer informação fornecida por você, na forma que ela acreditar que seja adequada, sem que incorra com isto em qualquer obrigação com você.

Portadores de Licenças deste programa que desejarem ter informações sobre ele para: (1) a troca de informações entre programas criados independentemente e outros programas (inclusive este), e (2) o uso mútuo de informações intercambiadas, devem entrar em contato com o:

Centro de Atendimento a Clientes IBM
Telefones: 0-800-784262 ou 0(XX)21 546-4646
Av. Pasteur, 138 / 146
Botafogo
22290-240, Rio de Janeiro - RJ
Brasil

Estas informações podem estar disponíveis, observadas as condições e os termos apropriados, incluindo, em alguns casos, o pagamento de uma taxa.

O programa licenciado descrito nesta informação e todo o material licenciado disponível para o mesmo são fornecidos pela IBM sob termos do IBM Customer Agreement, IBM International Program License Agreement, ou qualquer acordo entre nós.

Quaisquer dados de desempenho aqui contido foram determinados em um ambiente controlado. Portanto, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medidas podem ter sido feitas em sistemas em nível de desenvolvimento e não há garantias de que estas medidas serão as mesmas nos sistemas gerais disponíveis. Além disso, algumas medidas podem ter sido estimadas através de extrapolação. Os resultados reais podem variar. Os usuários deste documento devem verificar o dado aplicável para o seu ambiente específico.

Informações pertinentes a produtos não-IBM foram obtidas dos fornecedores daqueles produtos, suas comunicações ou anúncios publicados ou outras fontes disponíveis de publicidade. A IBM não testou aqueles produtos e não pode confirmar a precisão do desempenho, compatibilidade ou quaisquer outras reclamações relacionadas aos produtos não-IBM. Questões sobre as capacidades dos produtos não-IBM devem ser endereçadas aos fornecedores daqueles produtos.

Todas as instruções referentes a intenções ou ordens futuras da IBM estão sujeitas a alterações ou remoções sem qualquer aviso, e representam apenas metas e objetivos.

Estas informações podem conter exemplos de dados e relatórios utilizados em operações comerciais cotidianas. Para ilustrá-las o mais completamente possível, os exemplos incluem nomes de pessoas, empresas, marcas e produtos. Todos esses nomes são fictícios e qualquer semelhança com nomes e endereços utilizados por uma empresa comercial real é mera coincidência.

DIREITOS AUTORAIS:

Estas informações podem conter programas de aplicação de amostra em linguagem fonte, que ilustram técnicas de programação em várias plataformas operacionais. Você pode copiar, alterar e distribuir estes programas de amostra, de qualquer forma sem nenhum pagamento à IBM para os propósitos de desenvolvimento, uso, marketing ou distribuição dos programas de aplicação de acordo com a interface de programação de aplicação para a plataforma para a qual os programas de amostra foram escritos. Estes exemplos não foram profundamente testados sob todas as condições. A IBM, portanto, não pode garantir ou subentender confiabilidade, aproveitabilidade ou funcionamento destes programas.

Cada cópia ou qualquer porção destes programas de amostra ou qualquer trabalho derivativo deve incluir um aviso de copyright, como a seguir:

© (nome de sua empresa) (ano). Partes deste código são derivados dos Programas de Amostras da IBM Corp. © Copyright IBM Corp. _digite o ano ou os anos_. Todos os direitos reservados.

Marcas

Os seguintes termos, que podem estar destacados por um asterisco (*), são marcas da International Business Machines Corporation nos Estados Unidos da América e/ou em outros países.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

Os termos a seguir são marcas ou marcas de serviços de outras empresas:

Microsoft, Windows e Windows NT são marcas ou marcas de serviços da Microsoft Corporation.

Java ou todas as marcas e logotipos baseados em Java e Solaris são marcas da Sun Microsystems, Inc. nos Estados Unidos da América e/ou em outros países.

Tivoli e NetView são marcas da Tivoli Systems Inc. nos Estados Unidos da América e/ou em outros países.

UNIX é uma marca de serviço nos Estados Unidos da América e/ou outros países, e é licenciada exclusivamente através da X/Open Company Limited.

Outros nomes de companhias, produtos ou serviços, que podem ser denotados por um duplo asterisco (**) podem ser marcas ou serviços de terceiros.

Índice Remissivo

E

F

G

I

J

L

M

P

R

S

T

U

V

W

X

Y

Z

Comunicando-se com a IBM

Se o problema for técnico, revise e execute as ações sugeridas pelo *Troubleshooting Guide* antes de contactar o Suporte a Clientes DB2. Este guia sugere informações que você pode reunir para auxiliar o Suporte ao Cliente DB2 para servi-lo melhor.

Para obter informações ou para solicitar qualquer dos produtos DB2 Universal Database entre em contato com um representante IBM em uma filial local; ou entre em contato com qualquer revendedor de software IBM autorizado.

Se você mora no Brasil, ligue para o Centro de Atendimento a Clientes:

- 0-800-784-262
- 0(**)21 546-4646 para informar-se sobre as opções de serviço disponíveis.

Informações do Produto

Se você mora no Brasil, ligue para o Centro de Atendimento a Clientes:

- 0-800-784-262 para obter informações gerais.
- (019) 887-7591 - FAX para solicitar publicações.

http://www.ibm.com/software/data/
As páginas DB2 da World Wide Web oferecem informações sobre as novidades atuais do DB2, as descrições dos produtos, a programação educacional e muito mais.

http://www.ibm.com/software/data/db2/library/
O DB2 Product and Service Technical Library oferece acesso a questões freqüentes como dificuldades, manuais e informações técnicas atualizadas do DB2.

Nota: Estas informações podem estar somente em inglês.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
O site da Web de encomendas de Publicações Internacionais fornecem informações sobre como adquirir manuais.

http://www.ibm.com/education/certify/
O Programa de Certificação Profissional, do site Web da IBM fornece informações de teste de certificação para uma variedade de produtos IBM, inclusive do DB2.

ftp.software.ibm.com
Efetue logon como anonymous. No diretório `/ps/products/db2`, você

encontrará demonstrações, correções, informações e ferramentas relativos ao DB2 e vários outros produtos.

comp.databases.ibm-db2, bit.listserv.db2-l
Estes novos grupos da Internet estão disponíveis para usuários que queiram dividir suas experiências com produtos do DB2.

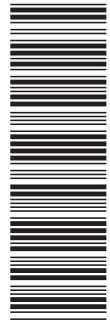
Em Compuserve: GO IBMDB2
Digite este comando para acessar os fóruns da Família DB2 IBM: Todos os produtos DB2 são suportados através destes fóruns.

Para obter informações sobre como entrar em contato com a IBM fora do Brasil, consulte o Apêndice A do *IBM Software Support Handbook*. Para acessar este documento, vá para a seguinte página da Web: http://www.ibm.com/support/, e então selecione o link IBM Software Support Handbook próximo ao rodapé da página.

Nota: Em alguns países, os distribuidores autorizados da IBM devem entrar em contato com sua estrutura de suporte de distribuição ao invés do Centro de Suporte IBM.

Número da Peça: CT7C0BP

Impresso em Brazil



CT7C0BP

S517-6993-00

Spine information:

IBM® DB2® Spatial Extender

DB2 Spatial Extender Referência e Guia do Usuário

Versão 7