IBM DB2 Extension Spatiale

Guide d'utilisation et de référence

Version 7

SC11-1684-00

IBM DB2 Extension Spatiale

Guide d'utilisation et de référence

Version 7

SC11-1684-00

Important

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à l'annexe «Remarques» à la page 333.

Réf. US : SC27-0701-00

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- http://www.fr.ibm.com (serveur IBM en France)
- http://www.can.ibm.com (serveur IBM au Canada)
- http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
Tour Descartes
92066 Paris-La Défense Cedex 50

Table des matières

Figures

Tableaux

Avis aux lecteurs canadiens

Le présent document a été traduit en France. Voici les principales différences et particularités dont vous devez tenir compte.

Illustrations

Les illustrations sont fournies à titre d'exemple. Certaines peuvent contenir des données propres à la France.

Terminologie

La terminologie des titres IBM peut différer d'un pays à l'autre. Reportez-vous au tableau ci-dessous, au besoin.

IBM France	IBM Canada
ingénieur commercial	représentant
agence commerciale	succursale
ingénieur technico-commercial	informaticien
inspecteur	technicien du matériel

Claviers

Les lettres sont disposées différemment : le clavier français est de type AZERTY, et le clavier français-canadien de type QWERTY.

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

- les pages de codes 850 (multilingue) et 863 (français-canadien),
- le code pays 002,
- le code clavier CF.

Nomenclature

Les touches présentées dans le tableau d'équivalence suivant sont libellées différemment selon qu'il s'agit du clavier de la France, du clavier du Canada ou du clavier des États-Unis. Reportez-vous à ce tableau pour faire

correspondre les touches françaises figurant dans le présent document aux touches de votre clavier.

France	Canada	Etats-Unis
↖ (Pos1)	↖	Home
Fin	Fin	End
⇞ (PgAr)	⇞	PgUp
⇟ (PgAv)	⇟	PgDn
Inser	Inser	Ins
Suppr	Suppr	Del
Echap	Echap	Esc
Attn	Intrp	Break
Impr écran	ImpEc	PrtSc
Verr num	Num	Num Lock
Arrêt défil	Défil	Scroll Lock
⇪ (Verr maj)	FixMaj	Caps Lock
AltGr	AltCar	Alt (à droite)

Brevets

Il est possible qu'IBM détienne des brevets ou qu'elle ait déposé des demandes de brevets portant sur certains sujets abordés dans ce document. Le fait qu'IBM vous fournisse le présent document ne signifie pas qu'elle vous accorde un permis d'utilisation de ces brevets. Vous pouvez envoyer, par écrit, vos demandes de renseignements relatives aux permis d'utilisation au directeur général des relations commerciales d'IBM, 3600 Steeles Avenue East, Markham, Ontario, L3R 9Z7.

Assistance téléphonique

Si vous avez besoin d'assistance ou si vous voulez commander du matériel, des logiciels et des publications IBM, contactez IBM direct au 1 800 465-1234.

Introduction

Le présent manuel est divisé en deux parties. La première contient des informations sur la conception de DB2 Extension Spatiale ; elle explique comment installer, configurer et administrer DB2 Extension Spatiale, mais aussi comment réaliser des programmes pour ce logiciel sous Windows NT et AIX. La seconde propose des informations de référence sur les procédures mémorisées, les géométries, les fonctions, messages et vues du catalogue que vous utilisez avec DB2 Extension Spatiale

A qui s'adresse ce manuel ?

Ce manuel est destiné aux administrateurs chargés de configurer l'environnement spatial et aux programmeurs responsables du développement des applications utilisant des données spatiales.

Conventions

Les conventions de mise en évidence suivantes sont utilisées dans le présent manuel :

En gras
: Sont indiqués les commandes ou les éléments de contrôle graphiques tels que les noms de zone, de dossier, d'icône ou d'option de menu.

`Par une police à espacement fixe`
: Sont illustrés les exemples de code ou de texte que vous devez entrer.

En italique
: Sont représentés les variables que vous devez remplacer par une valeur, les noms des manuels ou les termes importants.

EN MAJUSCULES
: sont indiqués les mots clés SQL et les nom d'objets (tables, vues, serveurs, etc.)

Commentaires et suggestions

Votre avis est important. Il nous aide à vous fournir des informations de qualité et de grande précision. Veuillez nous consacrer quelques instants pour nous communiquer votre opinion concernant le présent manuel ou tout autre document DB2. Vous pouvez nous faire parvenir vos commentaires par l'un des biais suivants :

- Via Internet. Vous trouverez un formulaire en ligne destiné à recueillir les commentaires du lecteur et proposé par IBM Data Management à l'adresse http://www.ibm.com/software/data/rcf.
- Par courrier électronique à l'adresse comments@vnet.ibm.com. Veillez à préciser le nom et le numéro de version du produit, ainsi que le nom et la référence du manuel. Si vos commentaires concernent un extrait de texte spécifique, indiquez son emplacement (chapitre, en-tête de section, numéro de tableau, numéro de page, titre de la rubrique d'aide, etc.).

Partie 1. Utilisation de DB2 Extension Spatiale

Chapitre 1. Présentation de DB2 Extension Spatiale

Le présent chapitre introduit DB2 Extension Spatiale. Il explique l'objectif de ce logiciel, décrit les données qu'il traite et donne des exemples de son utilisation. Il se termine par une présentation succincte du reste du manuel.

Objectif de DB2 Extension Spatiale

DB2 Extension Spatiale permet de créer un *système d'informations géographiques* (SIG) ; autrement dit, un ensemble complexe d'objets, de données et d'applications qui vous sert à générer et à analyser des informations spatiales sur des entités géographiques. Par *entités géographiques*, on entend les objets qui composent la surface de la terre et ceux qui l'occupent. Ils constituent à la fois l'environnement naturel (rivières, forêts, collines, déserts, etc.) et culturel (villes, maisons d'habitation, immeubles de bureaux, points de repère, etc.)

Les *informations spatiales* traitent de faits, tels que :

- L'emplacement des entités géographiques par rapport à leur environnement ; par exemple, l'implantation des hôpitaux et des cliniques dans une ville, ou la proximité d'habitations dans la ville par rapport aux zones sismiques locales.
- Les relations entre les différentes entités géographiques ; par exemple, des informations sur un système hydrographique spécifique dans une région déterminée ou sur certains ponts dans cette région qui enjambent les cours d'eau de ce système.
- Des mesures qui s'appliquent à une ou plusieurs entités géographiques ; par exemple, la distance entre un immeuble de bureaux et la limite de la parcelle correspondante ou le périmètre d'une réserve ornithologique.

Les informations spatiales, seules ou associées aux données d'un SGBD relationnel, peuvent vous orienter dans la conception de projets ou dans vos choix commerciaux ou stratégiques. Ainsi, supposons que le responsable d'un centre régional d'allocations familiales ait besoin de vérifier le nombre de personnes qui bénéficient d'une aide ou en sollicitent une, et qui habitent dans sa zone de compétence. DB2 Extension Spatiale peut déduire ces informations de l'emplacement de la zone couverte et des adresses des bénéficiaires et des demandeurs.

Ou encore, supposons que le propriétaire d'une chaîne de restaurants souhaite ouvrir des établissements dans des villes voisines. Pour déterminer l'emplacement de ses nouveaux restaurants, il a besoin de connaître la réponse aux questions suivantes : dans ces villes, où existe-t-il une

concentration de la clientèle qui fréquente normalement ses restaurants ? Où sont les principaux axes routiers ? Où le taux de criminalité est-il le plus faible ? Où sont situés les établissements concurrents ? DB2 Extension Spatiale permet d'afficher visuellement ces informations spatiales et le SGDB relationnel sous-jacent peut générer les intitulés et le texte expliquant les écrans.

Le présent manuel comporte d'autres exemples d'utilisation de DB2 Extension Spatiale, notamment dans le «Chapitre 7. Extraction et analyse d'informations spatiales» à la page 55, le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59, et le «Chapitre 14. Fonctions spatiales associées aux requêtes SQL» à la page 167.

Données représentant des entités géographiques

Cette section présente les données que vous générez, stockez ou manipulez pour obtenir des informations spatiales. Elle traite des sujets suivants :

- Représentation des entités géographiques par des données
- Nature des données spatiales
- Modes de génération des données spatiales

Représentation des entités géographiques par des données

Dans DB2 Extension Spatiale, une entité géographique peut être représentée par tout ou partie d'une ligne dans une table. Ainsi, prenons deux des entités géographiques mentionnées à la section «Objectif de DB2 Extension Spatiale» à la page 3, les immeubles de bureaux et les habitations. A la figure 1 à la page 5, chaque ligne de la table BRANCHES représente une succursale d'une banque. Par ailleurs, chaque ligne de la table CUSTOMERS (figure 1 à la page 5), prise dans sa totalité, représente un client de la banque. Toutefois, une partie de chaque ligne (celle qui contient l'adresse du client) peut être considérée comme représentant le domicile du dit client.

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

Figure 1. Ligne de table représentant une entité géographique ; ligne de table dont les données d'adresse représentent une entité géographique. La ligne de données de la table BRANCHES représente la succursale d'une banque. Les cellules correspondant aux données d'adresse de la table CUSTOMERS indiquent le domicile d'un client. Les noms et adresses mentionnés dans les deux tables sont fictifs.

Les tables de la figure 1 contiennent des données qui identifient et décrivent les succursales et les clients de la banque. Il s'agit de *données d'attribut*.

Un sous-ensemble des données d'attribut (celles qui indiquent les adresses des succursales et des clients) peut être converti en des valeurs qui fournissent des informations spatiales. Par exemple, dans la figure 1, l'une des succursales de la banque est située 92467 Airzone Blvd., San Jose CA 95141, et le domicile d'un client, 9 Concourt Circle, San Jose CA 95141. DB2 Extension Spatiale peut convertir ces adresses en valeurs qui indiquent l'emplacement de la succursale et du domicile du client par rapport à leur environnement. La figure 2 illustre les tables BRANCHES et CUSTOMERS comportant de nouvelles colonnes destinées à contenir ce type de valeurs.

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

Figure 2. Tables comportant des colonnes spatiales. Dans chaque table, la colonne LOCATION contient les coordonnées correspondant aux adresses.

Lorsque des adresses ou des identificateurs similaires servent de point de départ pour créer les informations spatiales, ils sont appelés *données source*. Les valeurs dérivées de ces données fournissent des informations spatiales et sont donc appelés *données spatiales*. La section suivante décrit les données spatiales et présente les types de données associés.

Nature des données spatiales

La plupart des données spatiales sont constituées de coordonnées. Une *coordonnée* est un nombre qui indique une position par rapport à un point de référence. Par exemple, les latitudes sont des coordonnées qui indiquent la position par rapport à l'équateur, et les longitudes, par rapport au méridien de Greenwich. Ainsi, la position du Parc national de Yellowstone (Etats-Unis) est définie par sa latitude (44,45 degrés au nord de l'équateur) et sa longitude (110,40 degrés à l'ouest du Méridien de Greenwich).

Les latitudes, les longitudes, leurs points de référence et autres paramètres associés sont désignés collectivement par l'expression *système de coordonnées*. Il existe également des systèmes de coordonnées qui sont fondés sur d'autres valeurs que la latitude et la longitude. Ces systèmes disposent de leurs propres mesures des positions, points de référence et paramètres distincts supplémentaires.

L'élément de données spatiale le plus simple consiste en deux coordonnées qui définissent la position d'un élément géographique distinct. (Un *élément de donnée* est la valeur indiquée dans une cellule de table relationnelle.) Un élément de donnée spatiale plus élaboré comprend plusieurs coordonnées qui définissent un chemin linéaire, tel qu'une route ou une rivière. Un troisième type consiste en coordonnées qui définissent le périmètre d'une zone ; par exemple, le bord d'une parcelle de terrain ou d'une plaine inondable. Ces types d'éléments de données, ainsi que d'autres, pris en charge par DB2 Extension Spatiale sont décrits de façon plus détaillée au «Chapitre 13. Géométries et fonctions spatiales associées» à la page 129.

Chaque élément de donnée spatiale est une instance d'un type de données spatiale. Le type de données de deux coordonnées qui désignent un emplacement est ST_Point, celui des coordonnées qui définissent des chemins linéaires est ST_LineString, et celui de celles définissant des périmètres ST_Polygon. Ces types, ainsi que d'autres associés aux données spatiales, sont des types structurés appartenant à une même hiérarchie. Pour une description de cette hiérarchie, reportez-vous à la section «Présentation des types de données spatiales» à la page 33.

Provenance des données spatiales

Vous pouvez obtenir des données spatiales par différentes méthodes :

- En les dérivant de données d'attribut
- En les dérivant d'autres données spatiales

- En les important

Utilisation de données d'attribut en tant que données source

DB2 Extension Spatiale peut dériver des données spatiales de données d'attribut telles que des adresses (comme mentionné à la section «Représentation des entités géographiques par des données» à la page 4). Ce processus s'appelle le *géocodage*. Pour comprendre la séquence impliquée, considérez la figure 2 à la page 5, comme la photo «avant», et la figure 3, comme la photo «après». Dans la figure 2 à la page 5, les tables BRANCHES et CUSTOMERS comportent toutes deux une colonne vide destinée à recevoir des données spatiales. Supposons que DB2 Extension Spatiale géocode les adresses contenues dans ces tables pour obtenir les coordonnées correspondantes et qu'il place ensuite ces dernières dans les colonnes. La figure 3, illustre le résultat de cette opération.

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

Figure 3. Tables comportant des données spatiales dérivées de données source. La colonne LOCATION de la table CUSTOMERS contient les coordonnées qu'un géocodeur a dérivé de l'adresse indiquée dans les colonnes ADDRESS, CITY, STATE et ZIP. Parallèlement, la colonne LOCATION de la table BRANCHES contient les coordonnées que le géocodeur a déduit de l'adresse indiquée dans les colonnes ADDRESS, CITY, STATE et ZIP de cette table. Il s'agit d'un exemple fictif ; les coordonnées indiquées ne sont pas réelles.

DB2 Extension Spatiale utilise une fonction appelée *géocodeur* pour convertir les données d'attribut en données spatiales et placer celles-ci dans les colonnes de la table. Pour plus d'informations sur les géocodeurs, reportez-vous à la section «Présentation du géocodage» à la page 41.

Utilisation d'autres données spatiales en tant que données source

Les données spatiales peuvent être générées à partir de données d'attribut, mais aussi à partir d'autres données spatiales. Ainsi, supposons que la banque dont les succursales sont définies dans la table BRANCHES veuille connaître le nombre de clients résidant dans un rayon d'environ 7 km de chaque succursale. Avant que la banque n'obtienne ces informations de la base de données, elle doit lui fournir la définition de la zone en question pour chaque succursale. Une fonction de DB2 Extension Spatiale, ST_Buffer, permet de créer ce type de définition. En utilisant les coordonnées de chaque succursale en tant que données d'entrée, ST_Buffer peut générer les coordonnées

permettant de délimiter les périmètres des zones souhaitées. La figure 4 , illustre la table BRANCHES contenant les informations fournies par la fonction 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

Figure 4. Table contenant les nouvelles données spatiales dérivées de données spatiales existantes. Les coordonnées de la colonne SALES_AREA ont été générées par la fonction ST_Buffer à partir des coordonnées figurant dans la colonne LOCATION. A l'instar des coordonnées de la colonne LOCATION, celles de la colonne SALES_AREA sont fictives.

Outre la fonction ST_Buffer, DB2 Extension Spatiale comporte plusieurs autres fonctions permettant de générer de nouvelles données spatiales à partir de données du même type existantes. Pour la description de ces fonctions dont ST_Buffer, reportez-vous à la section «Fonctions de génération de nouvelles géométries à partir de géométries existantes» à la page 156.

Importation de données spatiales

Une troisième méthode permet d'obtenir des données spatiales ; elle consiste à les importer à partir de fichiers existant dans un format pris en charge par DB2 Extension Spatiale. Pour la description de ces formats, reportez-vous au «Chapitre 16. Formats de fichiers pour données spatiales» à la page 307. Ces fichiers contiennent des données généralement utilisées pour des cartes : suivi du recensement, plaines inondables, failles sismiques, etc. En associant ces données à des données spatiales que vous avez créées, vous pouvez accéder à un plus grand nombre d'informations géographiques. Par exemple, si un service de travaux publics doit déterminer les risques auxquels est exposée une zone d'habitation, il peut utiliser la fonction ST_Buffer pour définir un périmètre autour de cette zone. Il peut ensuite importer des données concernant les plaines inondables et les failles sismiques pour voir si ces zones chevauchent la zone d'habitation.

Création et utilisation d'un SIG DB2 Extension Spatiale

Vous pouvez créer un SIG DB2 Extension Spatiale en configurant DB2 Extension Spatiale et en développant des projets SIG au sein des environnements combinés de DB2 Extension Spatiale et de son SGBD relationnel DB2 sous-jacent. Vous pouvez utiliser le SIG lors de la mise en oeuvre de ces projets ; autrement dit, en générant et en analysant les informations (spatiales et classiques) qu'ils sont destinés à vous fournir. La difficulté réside dans l'exécution des différents ensembles de tâches. Cette

section présente les interfaces qui vous permettent d'effectuer ces tâches, décrit celles-ci et fournit un exemple les illustrant.

Interfaces vers DB2 Extension Spatiale et ses fonctions associées

La présente section décrit brièvement les interfaces qui vous permettent de créer un SIG DB2 Extension Spatiale (définition des ressources correspondantes, obtention de données spatiales, etc.) et de l'utiliser (génération et analyse d'informations sur des entités géographiques).

Vous pouvez créer un SIG DB2 Extension Spatiale en appliquant l'une des méthodes suivantes :

- Utilisation des écrans et des options de menu DB2 Extension Spatiale accessibles à partir du centre de contrôle DB2. Pour les procédures correspondantes, reportez-vous aux chapitres suivants :
  - «Chapitre 3. Configuration des ressources» à la page 23
  - «Chapitre 4. Définition et enregistrement des colonnes spatiales, et activation d'un géocodeur» à la page 33
  - «Chapitre 5. Peuplement des colonnes spatiales» à la page 41
  - «Chapitre 6. Création d'index spatiaux» à la page 53
- Exécution d'un programme d'application qui appelle les procédures mémorisées DB2 Extension Spatiale. Pour concevoir ce type de programme, reportez-vous au «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.
- Utilisation du Centre de contrôle et d'un programme d'application. Par exemple, utilisez le Centre de contrôle pour appeler le géocodeur par défaut. Si vous souhaitez, en outre, recourir à un autre géocodeur, vous devez tout d'abord l'enregistrer auprès de DB2 Extension Spatiale en appelant la procédure mémorisée db2gse.gse_register_gc à partir d'un programme d'application. Pour plus d'informations sur les géocodeurs autres que celui défini par défaut, reportez-vous à la section «Présentation du géocodage» à la page 41. Pour plus d'informations sur la procédure mémorisée db2gse.gse_register_gc, reportez-vous à la section «db2gse.gse_register_gc» à la page 93.
- Utilisation du Centre de contrôle, d'un programme d'application ou des deux, associé(s) à d'autres interfaces. Par exemple, pour créer une table destinée à contenir des données générées par une fonction spatiale (un géocodeur, par exemple), vous pouvez utiliser les interfaces de l'interpréteur de commandes ou du Centre de contrôle.

Vous pouvez utiliser un SIG DB2 Extension Spatiale en procédant comme suit :

- Affichage des informations sous forme graphique à l'aide d'un navigateur géographique, tel qu'ArcExplorer conçu par l'Institut ESRI (Environmental Systems Research Institute).

- Soumission de requêtes SQL explicites à partir du Centre de contrôle ou de l'interpréteur de commandes DB2.
- Soumission de requêtes SQL à partir d'un programme d'application.

Tâches à exécuter pour créer et utiliser un SIG DB2 Extension Spatiale

La présente section décrit les tâches qui vous permettent de créer et d'utiliser un SIG DB2 Extension Spatiale. La procédure de création d'un SIG se divise en deux parties : la configuration de DB2 Extension Spatiale et le développement de projets SIG. Son utilisation consiste en la mise en oeuvre de projets. Cette présentation aborde donc en premier la configuration de DB2 Extension Spatiale, se poursuit avec le développement puis la mise en oeuvre d'un projet SIG. Enfin, la section se termine en indiquant dans quelle mesure les tâches mentionnées précédemment peuvent varier dans la pratique.

Configuration de DB2 Extension Spatiale

Pour configurer DB2 Extension Spatiale, procédez comme suit :

1. Planifiez et préparez les tâches inhérentes à cette configuration (choix des projets SIG à développer, de la base de données à activer pour DB2 Extension Spatiale, sélection du personnel chargé d'administrer DB2 Extension Spatiale et de développer les projets, etc.).
2. Installez DB2 Extension Spatiale.
3. Mettez en place les ressources nécessaire à la mise en oeuvre des projets SIG. Par exemple :

   Ressources fournies par DB2 Extension Spatiale
   Cela inclut un catalogue système, des types de données spatiales, des fonctions spatiales (dont un géocodeur par défaut), etc. On fait référence à cette tâche de définition des ressources par *activation de la base de données pour les opérations spatiales*.

   Géocodeurs conçus par des utilisateurs, des fournisseurs ou par les deux. Le géocodeur par défaut convertit les adresses aux Etats-Unis en données spatiales. Votre entreprise, entre autres, peut fournir des géocodeurs qui convertissent des adresses à l'étranger et d'autres types de données d'attribut en données spatiales.

Pour la procédure d'installation de DB2 Extension Spatiale, reportez-vous au «Chapitre 2. Installation de DB2 Extension Spatiale» à la page 17. Pour utiliser le Centre de contrôle dans le cadre de la mise en place des ressources, reportez-vous au «Chapitre 3. Configuration des ressources» à la page 23. Pour utiliser un programme d'application dans ce contexte, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59. Pour un exemple illustrant l'ensemble des tâches impliquées dans la configuration de DB2 Extension Spatiale, reportez-vous à la section «Un système permettant d'intégrer des données spatiales et classiques» à la page 13.

Développement et mise en oeuvre d'un projet SIG

Pour développer et mettre en oeuvre un projet SIG, procédez comme suit :

1. Planifiez et préparez les tâches correspondantes (définition des objectifs du projet, choix des tables et données souhaitées, détermination du ou des systèmes de coordonnées à utiliser, etc.).
2. Choisissez le ou les systèmes de références spatiales à utiliser. Normalement, les valeurs de coordonnées peuvent être des entiers positifs, des nombres négatifs et des nombres décimaux. Toutefois, DB2 Extension Spatiale doit enregistrer toutes ces valeurs sous forme de nombres entiers positifs. Un *système de références spatiales* est un ensemble de paramètres qui définit le mode de conversion des nombres négatifs et décimaux d'un système de coordonnées déterminé en nombres entiers positifs, et ce, en vue de leur stockage par DB2 Extension Spatiale. Après avoir choisi le système de coordonnées à utiliser pour une colonne spatiale, vous devez spécifier le système de références spatiales qui permettra d'effectuer la conversion appropriée sur ladite colonne. Si un système existant convient, utilisez-le, sinon, créez-en un autre.
3. Définissez une ou plusieurs colonnes destinées à recevoir les données spatiales, enregistrez-les auprès de DB2 Extension Spatiale et activez un géocodeur pour qu'elles soient mises à jour automatiquement.

   L'enregistrement d'une colonne spatiale consiste à l'enregistrer dans le catalogue DB2 Extension Spatiale. A partir du moment où vous avez effectué cette opération, la colonne est considérée comme une *couche*, car les informations qui en sont dérivées ajoutent une strate, ou couche, au paysage géographique virtuel que votre SIG crée. Après avoir enregistré la colonne, vous pouvez exécuter des opérations l'affectant ; ainsi, vous pouvez la peupler et y associer un index spatial.
4. Peuplez les colonnes spatiales :
   - Dans le cas d'un projet nécessitant un géocodeur, définissez les paramètres du géocodeur. Ensuite, exécutez-le afin qu'en une seule opération, il géocode toutes les données source disponibles et charge les coordonnées ainsi obtenues dans une couche.
   - Dans le cas d'un projet nécessitant l'importation de données spatiales, importez celles-ci.
5. Facilitez l'accès aux colonnes spatiales. Plus précisément, cela implique de définir des index qui permettent à DB2 d'accéder rapidement aux données spatiales, et des vues qui permettent aux utilisateurs d'extraire efficacement les données interdépendantes. Après avoir défini une vue de ce type, vous devez enregistrer les colonnes spatiales qu'elle contient en tant que couches.
6. Générez et analysez les informations spatiales et les informations commerciales associées. Cela implique d'analyser les colonnes spatiales et les colonnes d'attribut associées. Dans de telles requêtes, vous pouvez inclure des fonctions DB2 Extension Spatiale qui renvoient un large

éventail d'informations (distance la plus courte entre deux éléments géographiques, coordonnées définissant un périmètre autour d'une entité géographique, etc.). Pour plus d'informations sur la fonction ST_Buffer qui génère ce type de coordonnées, consultez les sections «Utilisation d'autres données spatiales en tant que données source» à la page 7, et «ST_Buffer» à la page 199. Pour des exemples de requêtes utilisant des fonctions spatiales, reportez-vous au «Chapitre 7. Extraction et analyse d'informations spatiales» à la page 55, et au «Chapitre 14. Fonctions spatiales associées aux requêtes SQL» à la page 167.

Pour utiliser le Centre de contrôle dans le cadre du développement d'un projet SIG, reportez-vous aux chapitres suivants :

- «Chapitre 3. Configuration des ressources» à la page 23
- «Chapitre 4. Définition et enregistrement des colonnes spatiales, et activation d'un géocodeur» à la page 33
- «Chapitre 5. Peuplement des colonnes spatiales» à la page 41
- «Chapitre 6. Création d'index spatiaux» à la page 53

Pour utiliser le Centre de contrôle dans le cadre de la mise en oeuvre d'un projet SIG, reportez-vous au «Chapitre 7. Extraction et analyse d'informations spatiales» à la page 55.

Pour utiliser un programme d'application dans le cadre du développement et de la mise en oeuvre d'un projet SIG, reportez-vous au «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Pour un exemple illustrant l'ensemble des tâches impliquées dans le développement et la mise en oeuvre d'un projet SIG, reportez-vous à la section «Projet d'implantation de succursales et d'ajustement des primes» à la page 14.

Variations possibles des tâches

Le contenu et l'ordre des ensembles de tâches que vous exécutez pour créer et utiliser un SIG DB2 Extension Spatiale peuvent varier en fonction de vos besoins et des interfaces auxquelles vous avez recours. Prenons, par exemple, les tâches suivantes : définition de colonnes en vue de l'obtention de données spatiales, enregistrement de ces colonnes en tant que couches et activation d'un géocodeur pour leur mise à jour automatique. Avec le Centre de contrôle, vous pouvez effectuer ces tâches conjointement à partir d'une même fenêtre. Si vous appelez des procédures mémorisées provenant d'un programme, vous pouvez les réaliser séparément et programmer leur exécution à votre gré.

Scénario : mise à jour du SIG d'une compagnie d'assurance

La présente section développe un scénario qui illustre les ensembles de tâches décrits dans la section précédente.

L'environnement des systèmes d'information de la compagnie d'assurance Safe Harbor Real Estate comporte un système DB2 Universal Database et un système de gestion de bases de données SIG. Dans une certaine mesure, des requêtes permettent d'extraire des combinaisons de données provenant des deux systèmes. Par exemple, une table DB2 stocke les informations sur les revenus et une table SIG l'emplacement des succursales de la société. Ainsi, il est possible de déterminer où sont localisées les succursales qui génèrent des revenus de montants déterminés. Toutefois, les données des deux systèmes ne peuvent pas être intégrées (les utilisateurs ne peuvent pas joindre des colonnes DB2 et des colonnes SIG) et les services DB2 tels que l'optimisation des requêtes ne peuvent pas être utilisés par le SIG. Pour remédier à cela, Safe Harbor a acquis le logiciel DB2 Extension Spatiale et créé un nouveau service de développement SIG. Les sections ci-après décrivent comment le service a configuré DB2 Extension Spatiale et réalisé son premier projet.

Un système permettant d'intégrer des données spatiales et classiques

Pour configurer DB2 Extension Spatiale, le service de développement SIG procéde ainsi :

1. Le service prépare l'intégration de DB2 Extension Spatiale dans son environnement DB2. Par exemple :
   a. L'équipe chargée de diriger le service embauche une équipe responsable de l'administration spatiale pour l'installation et la mise en oeuvre de DB2 Extension Spatiale, ainsi qu'une équipe chargée de l'analyse spatiale pour la génération et l'analyse des informations spatiales.
   b. Les décisions commerciales de la compagnie Safe Harbor reposant essentiellement sur les exigences de la clientèle, l'équipe de direction a donc décidé d'installer DB2 Extension Spatiale dans la base de données qui contient les informations concernant les clients. La plupart de ces informations sont stockées dans la table CUSTOMERS.

      Les membres du service de développement SIG ont trouvé plus pratique de se référer à la base de données sélectionnée en l'appelant *base de données SIG*. Ils sont cependant conscients qu'elle n'est pas uniquement réservée aux projets SIG et les applications non spatiales peuvent continuer à l'utiliser, comme auparavant.
2. L'équipe d'administration spatiale installe DB2 Extension Spatiale.

3. L'équipe d'administration spatiale définit les ressources nécessaires aux projets SIG :
   - L'équipe utilise le Centre de contrôle pour fournir les ressources qui activent la base de données SIG pour les opérations spatiales. Cela inclut un catalogue système, des types de données spatiales, des fonctions spatiales, etc.
   - La compagnie Safe Harbor a commencé à étendre son activité au Canada, l'équipe d'administration spatiale a donc contacté des fournisseurs de géocodeurs canadiens qui permettent de convertir les adresses canadiennes en données spatiales.

Projet d'implantation de succursales et d'ajustement des primes

Pour réaliser son premier projet SIG sous DB2 Extension Spatiale, le service de développement SIG procède comme suit :

1. Le service prépare la conception du projet, par exemple :
   - L'équipe de direction définit les objectifs du projet :
     - Déterminer où implanter les nouvelles succursales.
     - Moduler les primes en fonction de la proximité des clients par rapport aux zones de risque (zones ayant des taux d'accidents de la route ou de criminalité élevé, zones inondables, zones sismiques, etc.).
   - Le projet SIG est destiné aux clients et aux succursales aux Etats-Unis. C'est pourquoi l'équipe d'administration spatiale prend les décisions suivantes :
     - Seront utilisés des systèmes de coordonnées qui définissent de manière précise les régions des Etats-Unis où la compagnie Safe Harbor exerce son activité.
     - Le géocodeur par défaut sera utilisé car il est conçu pour le géocodage des adresses aux Etats-Unis.
   - L'équipe d'administration spatiale sélectionne les données nécessaires pour réaliser les objectifs du projet, ainsi que les tables à peupler.
2. Via le Centre de contrôle, l'équipe d'administration spatiale crée deux systèmes de références spatiales. L'un détermine comment convertir les coordonnées définissant l'emplacement des succursales, et l'autre, les coordonnées définissant le domicile des clients, en éléments de données stockables par DB2 Extension Spatiale.
3. A l'aide du Centre de contrôle, l'équipe d'administration spatiale définit les colonnes destinées à contenir les données spatiales, les enregistre en tant que couches et active un géocodeur pour leur mise à jour automatique.

- L'équipe ajoute une colonne LOCATION à la table CUSTOMERS qui contient déjà les adresses des clients. Le géocodeur par défaut convertira ces dernières en données spatiales qu'il chargera dans la colonne LOCATION.
- L'équipe crée une table OFFICES destinée à contenir les données qui sont pour le moment stockées dans un autre SIG. Il s'agit des adresses des succursales de la compagnie Safe Harbor, des données spatiales qui en sont dérivées par un géocodeur et des données spatiales qui définissent une zone d'un rayon d'environ 7 km autour de chaque succursale. Les données générées par le géocodeur peupleront la colonne LOCATION, et celles définissant les zones, la colonne SALES_AREA.
- L'équipe enregistre les colonnes LOCATION et SALES_AREA en tant que couches.
- L'équipe active le géocodeur par défaut afin que celui-ci mette à jour automatiquement les deux colonnes LOCATION.

4. L'équipe d'administration spatiale peuple ensuite la colonne LOCATION de la table CUSTOMERS, la totalité de la table OFFICES ainsi qu'une nouvelle table appelée HAZARD_ZONES.
   - L'équipe utilise le Centre de contrôle pour peupler la colonne LOCATION de la table CUSTOMERS.
     a. L'équipe demande au géocodeur d'insérer les données spatiales correspondant à une adresse dans la colonne LOCATION uniquement dans le cas suivant : l'adresse en question doit être strictement identique à son équivalent figurant dans les registres du bureau de recensement des Etats-Unis. [DB2 Extension Spatiale est accompagné d'un fichier d'adresses fournies par le bureau de recensement. Avant que le géocodeur ne convertisse une adresse provenant des données source en données spatiales, il essaye de trouver son équivalent dans le fichier. Les utilisateurs doivent indiquer le degré de précision (exprimé en pourcentage) requis pour que les données spatiales soient transférées dans une table. Ce pourcentage s'appelle une *précision*.]
     b. L'équipe exécute le géocodeur en traitement par lots afin qu'il géocode toutes les adresses de la table en une seule opération. A la consternation générale, le géocodeur rejette environ une adresse sur dix !
     c. L'équipe présume alors qu'il s'agissait de nouvelles adresses dont l'équivalent exact ne figurait pas encore dans les registres du bureau de recensement. Pour remédier à ce problème, la précision a été réduite à 85.
     d. L'équipe exécute de nouveau le géocodeur en traitement par lots. Le taux de rejet des adresses tombe alors à un niveau acceptable.

- A l'aide d'un utilitaire fourni par l'autre SIG, l'équipe charge les données sur les succursales dans un fichier et, à partir du Centre de contrôle, elle les importe ensuite du fichier dans la nouvelle table OFFICES.
- A l'aide du Centre de contrôle, l'équipe crée une table HAZARD_ZONES, enregistre les colonnes spatiales la composant en tant que couches, puis elle y importe les données. Celles-ci proviennent d'un fichier fourni par une société de cartographie.

5. Via le Centre de contrôle, l'équipe facilite l'accès aux nouvelles couches de la manière suivante :
   - Création d'index associés aux couches.
   - Création d'une vue qui regroupe les colonnes des tables CUSTOMERS et HAZARD_ZONES. L'équipe a ensuite enregistré les colonnes spatiales de la vue en tant que couches.
6. L'équipe d'analyse spatiale exécute ensuite des requêtes afin d'obtenir les informations susceptibles de l'aider à remplir les objectifs fixés au départ, à savoir : déterminer où implanter les nouvelles succursales et moduler les primes d'assurance en fonction de la proximité des clients par rapport aux zones de risques.

Chapitre 2. Installation de DB2 Extension Spatiale

Le présent chapitre explique comment installer DB2 Extension Spatiale, et aborde les sujets suivants :

Configuration de DB2 Extension Spatiale

Un système DB2 Extension Spatiale est constitué d'une base de données DB2 Universal Database, du logiciel DB2 Extension Spatiale et d'un navigateur géographique (par exemple, ArcExplorer). Normalement, une base de données activée pour des opérations spatiales réside sur le serveur. Vous utilisez les applications client pour accéder aux données spatiales via les procédures mémorisées et les requêtes spatiales de DB2 Extension Spatiale. En outre, vous pouvez visualiser ces données via un navigateur géographique.

La figure 5 illustre l'architecture de DB2 Extension Spatiale.

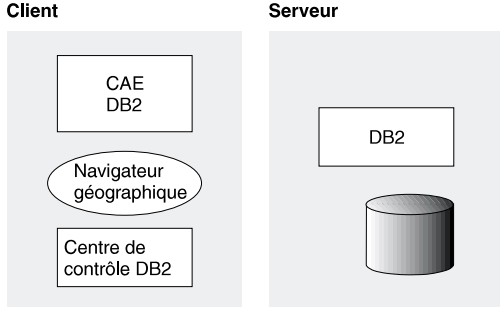


Figure 5. Configuration client-serveur

Configuration système requise

La présente section décrit la configuration matérielle et logicielle requise pour DB2 Extension Spatiale.

Systèmes d'exploitation pris en charge

DB2 Extension Spatiale peut être installé sur les systèmes d'exploitation suivants :

- AIX version 4.2 ou suivante
- Windows NT version 4.0 ou suivante avec Service Pack 5

Logiciel de base de données requis

Avant d'installer DB2 Extension Spatiale, vous devez avoir installé et configuré le logiciel DB2 sur votre système. Le tableau 1 répertorie les logiciels de base de données requis pour les composants DB2 Extension Spatiale client et serveur.

Tableau 1. Configuration logicielle minimale

Composant	Logiciel
Client	DB2 Administration Client, version 7.1[1]
Serveur	Au choix : • DB2 Universal Database Enterprise Edition, version 7.1 • DB2 Universal Database Enterprise – Extended Edition, version 7.1[2]
	Remarques : 1. Si vous *n'envisagez pas* d'utiliser le Centre de contrôle DB2, le navigateur géographique pour l'accès aux données spatiales ou le programme exemple DB2 Extension Spatiale, vous pouvez utiliser une version précédente de DB2 Administration Client. 2. Bien que vous puissiez utiliser DB2 Extension Spatiale avec DB2 Universal Database Enterprise - Extended Edition, les index spatiaux ne peuvent pas être répartis entre plusieurs noeuds comme en environnement MPP (Massive Parallel Processing).

Espace disque requis

Le tableau 2 indique l'espace disque recommandé pour DB2 Extension Spatiale.

Tableau 2. Espace disque requis

Composant DB2 Extension Spatiale	Espace disque
Bibliothèque serveur DB2 Extension Spatiale (comprenant la bibliothèque serveur, les données de référence du géocodeur et la documentation DB2 Extension Spatiale)	600 Mo
Support client DB2 Extension Spatiale (comprenant les données du programme exemple)	15 Mo

Installation de DB2 Extension Spatiale

La présente section fournit les informations nécessaires à l'installation de DB2 Extension Spatiale sous Windows et Aix.

Avant de commencer

Si ce n'est déjà fait, installez le composant DB2 Administration Client (outils d'administration dont le Centre de contrôle et le composant Run-Time Client) sur le poste de travail client, ainsi que DB2 Universal Database Enterprise Edition ou DB2 Universal Database Enterprise - Extended Edition. Vous trouverez une description des procédures correspondantes dans le manuel *Mise en route* approprié.

Installation de DB2 Extension Spatiale sous Windows NT

Pour installer DB2 Extension Spatiale sous Windows NT, procédez comme suit :

1. Connectez-vous au système sous un ID utilisateur doté des droits d'administration requis.
2. Fermez tous les autres programmes.
3. Insérez le CD-ROM dans l'unité. Le tableau de bord de l'installation s'affiche.
4. Facultatif : Cliquez sur **Notes de mise à jour** pour vérifier si la procédure d'installation a été modifiée dans les remarques relatives à l'installation de DB2 Extension Spatiale, puis revenez au tableau de bord de DB2 Extension Spatiale.
5. Cliquez sur **Installer**.
6. Répondez aux invites du programme d'installation. Vous pouvez utiliser l'aide en ligne pour vous assister dans la suite des opérations. Pour ce faire, cliquez sur **Aide** ou appuyez sur la touche F1.

Une fois l'installation terminée, DB2 Extension Spatiale réside dans le répertoire %DB2PATH% (par exemple, c:\sqllib).

Installation de DB2 Extension Spatiale sous AIX

Pour installer DB2 Extension Spatiale sous AIX, procédez comme suit :

1. Connectez-vous en tant qu'utilisateur root.
2. Insérez le CD-ROM dans l'unité.
3. Montez le CD-ROM sur votre système AIX. Pour plus d'informations sur cette procédure, reportez-vous au manuel *IBM DB2 Universal Database pour UNIX - Mise en route*.
4. Placez-vous dans le répertoire où le CD-ROM est monté en lançant la commande suivante :

   ```
   cd /cdrom
   ```

   où */cdrom* désigne le point de montage de l'unité de CD-ROM sous AIX.

5. Entrez la commande **db2setup** pour démarrer le programme d'installation de DB2. La fenêtre d'installation de DB2 Extension Spatiale s'affiche.

   Remarque : Le programme d'installation de DB2 démarre lentement car il recherche des informations sur votre système.

6. Dans la liste des produits figurant sur l'écran d'installation de DB2 Extension Spatiale, sélectionnez ceux que vous souhaitez installer, puis cliquez sur **OK**.

   Pour plus d'informations sur la procédure d'installation de DB2 Extension Spatiale, cliquez sur **Aide**.

Une fois l'installation terminée, DB2 Extension Spatiale réside dans le répertoire /usr/lpp/db2_07_01.

Vérification de l'installation

Une fois l'installation de DB2 Extension Spatiale terminée, vous pouvez vérifier qu'elle a été effectuée correctement à l'aide du programme exemple. Avant d'exécuter celui-ci, vous devez le rendre exécutable et créer la base de données exemple.

Remarque : Veillez à utiliser le compilateur spécifié dans le fichier makefile de DB2 Extension Spatiale.

Pour compiler et exécuter le programme exemple destiné à Windows NT, procédez comme suit :

1. Connectez-vous sous un ID utilisateur doté des privilèges administrateur.
2. A l'invite du système, entrez la commande **db2sampl** pour créer la base de données DB2 SAMPLE.
3. A l'invite du système, entrez la commande suivante :

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

Remarque : Pour réaliser l'étape 3 et continuer à vérifier l'installation, vous devez être propriétaire de l'instance DB2 par défaut (DB2–DB2).

4. Entrez **make rungsedemo**.
5. Entrez **rungsedemo.exe**.
6. Vérifiez les messages d'erreur et d'aboutissement qui s'affichent pendant l'exécution du programme.

Pour compiler et exécuter le programme exemple destiné à AIX, procédez comme suit :

1. Connectez-vous en tant qu'utilisateur root.
2. Créez ou mettez à jour une instance DB2.

3. A l'invite du système, entrez la commande **db2sampl** pour créer la base de données DB2 SAMPLE.
4. A l'invite du système, entrez la commande suivante :

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

   Remarque : Pour réaliser l'étape 4 et continuer à vérifier l'installation, vous devez être propriétaire de l'instance DB2 que vous créez ou que vous mettez à jour.
5. Entrez **make rungsedemo**.
6. Entrez **rungsedemo**.
7. Vérifiez les messages d'erreur et d'aboutissement qui s'affichent pendant l'exécution du programme.

Pour plus d'informations sur les programmes exemples, reportez-vous au «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Etapes de post-installation

Une fois DB2 Extension Spatiale correctement installé, vous devez envisager les étapes suivantes :

- Téléchargement d'ArcExplorer
- Exécution de l'utilitaire de mise à jour des instances DB2

Téléchargement d'ArcExplorer

IBM distribue le logiciel ArcExplorer Java 3.0 en tant que modèle de programme ; vous pouvez également l'obtenir en vous connectant au site Web de la société ESRI à l'adresse http://www.esri.com.

Pour plus d'informations sur l'installation et l'utilisation du logiciel ArcExplorer, reportez-vous au manuel *Using ArcExplorer*, également disponible sur le site Web ESRI.

ArcExplorer requiert la version 1.2.2 de Java 2 Runtime Environment (Standard Edition ou Enterprise Edition), que vous pouvez vous procurer gratuitement sur le site Web Sun à l'adresse http://java.sun.com.

Important : DB2 Universal Database version 7.1 est fourni avec le logiciel IBM JDK 1.1.8. Lorsque vous installez JRE 1.2.2 pour ArcExplorer, ne le placez pas dans le répertoire de DB2. N'oubliez pas définir la variable d'environnement CLASSPATH en conséquence.

Exécution de l'utilitaire de mise à jour des instances DB2 (db2iupdt)

L'utilitaire db2iupdt met à jour une instance DB2 spécifiée pour que :

- l'instance puisse appliquer une nouvelle configuration système.

- l'instance puisse accéder à une fonction associée à l'installation ou au retrait de certaines options de produit.

Sous AIX, cet utilitaire se trouve dans le répertoire /usr/lpp/db2_07_01. Si vous avez besoin d'aide, tapez `db2iupdt -h` sur la ligne de commande pour ouvrir un menu d'aide. Sous Windows NT, db2iupdt se trouve dans le répertoire \sqllib\bin. Placez-vous dans ce répertoire pour taper la commande. Pour une description complète de cette commande, reportez-vous au manuel *IBM DB2 Universal Database Command Reference*.

Et ensuite ?

Une fois DB2 Extension Spatiale installé, vous pouvez utiliser le Centre de contrôle de DB2 pour configurer l'environnement SIG et commencer à travailler avec des informations spatiales.

Pour appeler DB2 Extension Spatiale à partir du Centre de contrôle, procédez comme suit :

1. Dans la fenêtre du Centre de contrôle, développez l'arborescence des objets jusqu'à ce que vous trouviez le dossier **Bases de données** sous le serveur sur lequel doit s'exécuter DB2 Extension Spatiale.
2. Cliquez sur le dossier **Bases de données**. La liste des bases de données s'affiche dans le panneau de contenu figurant dans la partie droite de la fenêtre.
3. Cliquez avec le bouton droit de la souris sur la base de données que vous voulez utiliser, puis, dans le menu en incrustation, cliquez sur l'opération spatiale à exécuter.

Pour plus d'informations sur l'utilisation de DB2 Extension Spatiale à partir du Centre de contrôle, reportez-vous aux chapitres suivants :

- «Chapitre 3. Configuration des ressources» à la page 23
- «Chapitre 4. Définition et enregistrement des colonnes spatiales, et activation d'un géocodeur» à la page 33
- «Chapitre 5. Peuplement des colonnes spatiales» à la page 41
- «Chapitre 6. Création d'index spatiaux» à la page 53

Chapitre 3. Configuration des ressources

Après avoir installé DB2 Extension Spatiale, vous êtes prêt à doter votre base de données des ressources nécessaires à la création des colonnes spatiales et à la manipulation de données spatiales. Le présent chapitre répertorie ces ressources et décrit deux des tâches qui vous permettent de les rendre disponibles : l'activation de la base de données pour les opérations spatiales et la création d'un système de références spatiales.

Inventaire des ressources

Les ressources auxquelles vous recourez pour créer des colonnes spatiales et manipuler des données spatiales sont les suivantes :

- *Données de référence* : adresses consultées par DB2 Extension Spatiale pour vérifier les adresses à géocoder.
- Ressources qui activent une base de données pour les opérations spatiales : procédures mémorisées, fonctions spatiales, etc.
- Géocodeurs autres que celui par défaut, provenant d'utilisateurs et de fournisseurs
- Systèmes de références spatiales

La présente section traite des ressources qui activent une base de données pour les opérations spatiales. Pour plus d'informations sur les géocodeurs non définis par défaut, reportez-vous à la section «Présentation du géocodage» à la page 41. Pour plus d'informations sur les systèmes de références spatiales, reportez-vous à la section «Présentation des systèmes de références spatiales et de coordonnées» à la page 25.

Données de référence

Les *Données de référence* correspondent aux adresses les plus récentes aux États-Unis collectées par le bureau de recensement fédéral. Avant que le géocodeur par défaut puisse convertir une adresse de votre base de donnée en coordonnées, tout ou partie de celle-ci doit correspondre à l'adresse figurant dans les données de référence.

Vous pouvez utiliser les données de référence à partir du moment où vous installez DB2 Extension Spatiale. Pour connaître l'espace disque requis pour ces données, reportez-vous à la section «Espace disque requis» à la page 18. Sous AIX, pour vérifier si les données ont été chargées correctement, regardez si elles se trouvent dans le répertoire

$DB2INSTANCE/sqllib/gse/refdata/.Sous Windows NT, pour vérifier si les données ont été chargées correctement, regardez si elles se trouvent dans le répertoire %DB2PATH%\gse\refdata\.

Ressources activant une base de données pour les opérations spatiales

La première tâche à effectuer après avoir installé DB2 Extension Spatiale consiste à activer la base de données pour les opérations spatiales. Cela signifie que vous devez déclencher le chargement de la base de données par DB2 Extension Spatiale avec les ressources suivantes :

- Procédures mémorisées. Lorsque vous demandez une action à partir du Centre de contrôle, DB2 Extension Spatiale appelle l'une des procédures mémorisées afin qu'elle exécute cette action.
- Types de données spatiales. Vous devez affecter un type de données spatiales à chaque table ou colonne de vue, destinée au stockage de données spatiales. Pour plus d'informations, reportez-vous à la section «Présentation des types de données spatiales» à la page 33.
- Vues et tables du catalogue DB2 Extension Spatiale. Certaines opérations dépendent du catalogue DB2 Extension Spatiale. Par exemple, avant qu'on puisse peupler une colonne associée à un type de données spatiales, celle-ci doit être enregistrée en tant que couche dans le catalogue. Pour plus d'informations sur les couches, reportez-vous à la section «Développement et mise en oeuvre d'un projet SIG» à la page 11
- Type d'index spatial. Il vous permet de définir les index associés aux couches.
- Fonctions spatiales. Vous pouvez y recourir pour utiliser des données spatiales de diverses manières ; par exemple, pour déterminer les relations entre différents éléments topographiques et générer d'autres données spatiales. Le géocodeur par défaut fait partie de ces fonctions. Il convertit les adresses aux Etats-Unis en coordonnées et insère ces dernières dans des colonnes spatiales. Pour plus d'informations sur les fonctions spatiales, reportez-vous au «Chapitre 13. Géométries et fonctions spatiales associées» à la page 129, et au «Chapitre 14. Fonctions spatiales associées aux requêtes SQL» à la page 167. Pour plus d'informations sur le géocodeur par défaut, reportez-vous à la section «Présentation du géocodage» à la page 41.
- Le schéma DB2GSE qui contient les objets mentionnés précédemment.

Pour déclencher le chargement de ces ressources à partir du Centre de contrôle, reportez-vous à la section «Activation d'une base de données pour les opérations spatiales» à la page 25. Pour exécuter la même tâche à l'aide d'une routine d'un programme d'application, reportez-vous au «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Activation d'une base de données pour les opérations spatiales

Afin de connaître les droits liés à l'activation d'une base de données pour les opérations spatiales, reportez-vous à la section «Classe d'autorisation» à la page 80.

Pour activer une base de données pour les opérations spatiales à partir du Centre de contrôle, procédez comme suit :

1. Dans la fenêtre du Centre de contrôle, développez l'arborescence des objets jusqu'à ce que vous trouviez le dossier **Bases de données** sous le serveur sur lequel doit s'exécuter DB2 Extension Spatiale.
2. Cliquez sur le dossier **Bases de données**. La liste des bases de données s'affiche dans le panneau de contenu figurant dans la partie droite de la fenêtre.
3. Cliquez avec le bouton droit de la souris sur la base de données de votre choix, puis sélectionnez **Extension Spatiale** —> **Activation** dans le menu en incrustation. DB2 Extension Spatiale fournit à la base de données les ressources qui vous permettront de créer et d'utiliser des colonnes et des données spatiales.

Rappel : Avant d'activer une base de données pour des opérations spatiales, vous devez avoir installé DB2 Extension Spatiale sur le serveur sur lequel réside la base de données.

Création d'un système de références spatiales

La présente section décrit les relations existant entre les systèmes de références spatiales et les systèmes de coordonnées. Elle explique également comment créer un système de références spatiales à partir du Centre de contrôle.

Présentation des systèmes de références spatiales et de coordonnées

Cette section poursuit la présentation des systèmes de coordonnées amorcée à la section «Nature des données spatiales» à la page 6 ; elle développe ensuite la définition des systèmes de références spatiales énoncée à la section «Développement et mise en oeuvre d'un projet SIG» à la page 11. Enfin, elle fournit des directives pour déterminer les valeurs à affecter aux paramètres d'un système de références spatiales.

Systèmes de coordonnées, coordonnées et mesures

Vous pouvez vous représenter un système de coordonnées comme une grille imaginaire qui couvre une zone géographique déterminée. Par exemple, une grille qui recouvre la terre, un pays ou une région. Chaque entité géographique de la zone est située à l'intersection d'un axe est-ouest et d'un axe nord-sud. Une valeur, appelée *abscisse (coordonnée X)*, indique la position de l'entité sur l'axe est-ouest et une autre valeur, appelée *ordonnée*

(coordonnée Y), indique la position sur l'axe nord-sud. Ces deux valeurs référencent la position de l'entité par rapport au centre de la grille ou *point d'origine*.

Les coordonnées X et Y du point d'origine ont toutes deux la valeur zéro. En allant du point d'origine vers l'est, les abscisses sont positives et en allant du point d'origine vers l'ouest, elles sont négatives. De la même manière, en allant du point d'origine vers le nord, les ordonnées sont positives et en allant du point d'origine vers le sud, elles sont négatives. Pour illustrer cette répartition, considérons l'exemple suivant : le système de coordonnées A comprend une grille qui couvre une large zone métropolitaine. Une valeur d'abscisse de 7 indiquerait un endroit situé à 7 unités de mesure à l'est du point d'origine de cette grille et une valeur d'abscisse de -9,5, un endroit situé à 9 unités et demie de mesure à l'ouest du dit point.

Chaque élément de données d'une colonne spatiale inclut (1) une abscisse (coordonnée X) et une ordonnée (coordonnée Y) qui définit la position d'une entité géographique ou (2) plusieurs coordonnées X et Y qui définissent l'emplacement des parties d'une entité ou la zone que couvre la dite entité. Il existe deux autres types de valeur (*coordonnée Z* et *mesure*) qui peuvent également figurer dans ces colonnes. Contrairement aux abscisses et aux ordonnées, les coordonnées Z et les mesures ne sont pas utilisées dans DB2 Extension Spatiale pour définir des positions ou des zones. Elles servent plutôt à transmettre des informations requises par une application SIG. Une coordonnée Z indique l'altitude ou la profondeur d'une entité géographique. Les coordonnées Z situées au-dessus du point d'origine sont positives, celles situées en-dessous sont négatives. Une mesure est une valeur numérique, elle peut transmettre n'importe quel type d'information. Par exemple, supposons que vous représentiez des puits de pétrole sur votre SIG. Si vous demandez à vos applications de traiter des valeurs qui désignent les ID d'épicentres associés à des données sismiques, vous pouvez stocker ces valeurs en tant que mesures.

Systèmes de références spatiales, décalages et facteurs d'échelle

Comme mentionné à la section «Systèmes de coordonnées, coordonnées et mesures» à la page 25, des coordonnées peuvent être négatives et représentées sous une forme décimale. C'est également le cas pour les mesures. Toutefois, pour réduire le surcroît d'occupation de la mémoire, DB2 Extension Spatiale stocke chaque coordonnée et mesure sous la forme d'un nombre entier non négatif (entier positif ou nul). Par conséquent, toutes les coordonnées négatives, décimales ainsi que les mesures doivent être converties en entiers non négatifs, afin que DB2 Extension Spatiale puisse les stocker. En outre, vous devez indiquer à DB2 Extension Spatiale le mode de conversion à utiliser en définissant certains paramètres. L'ensemble des valeurs de

paramètres à utiliser pour la conversion des coordonnées et des mesures dans une zone géographique déterminée constitue ce qu'on appelle un *système de références spatiales*.

Vous pouvez créer un système de références spatiales en procédant comme suit :

- Déterminez les coordonnées et les mesures négatives les plus petites associées aux entités que vous représentez. (Plus une valeur négative s'éloigne de zéro, plus elle est petite. Une abscisse (coordonnée X) de –10 est inférieure à une abscisse de –5, de même qu'une mesure de –100 est inférieure à –50.
- Spécifiez des *facteurs de décalage* (ou en abrégé *décalages*). Il s'agit de valeurs qui, lorsqu'on les soustrait de coordonnées et mesures négatives, donnent des nombres non négatifs.
- Spécifiez des *facteurs d'échelle*. Il s'agit de valeurs qui, lorsqu'elles sont multipliées par des coordonnées ou des mesures décimales, donnent des entiers dont la précision est au moins égale à celle de ces coordonnées ou mesures. Par exemple, prenons une coordonnée ayant une précision de 4 : 92,77. Vous pouvez la multiplier par une échelle de 100 et ainsi obtenir un entier doté de la même précision (4) : 9277.

Détermination des coordonnées et mesures négatives les plus petites

Avant de définir les paramètres d'un système de références spatiales, vous devez déterminer quelles sont l'abscisse (X), l'ordonnée (Y), la coordonnée Z et la mesure les plus petites dans la zone géographique contenant les entités sur lesquelles vous désirez des informations. Vous pouvez trouver ces valeurs en répondant aux questions suivantes :

- Parmi les entités que vous représentez, y-en-a-t-il une qui se situe à l'ouest du point d'origine du système de coordonnées que vous utilisez ? Si tel est le cas, quelle abscisse (coordonnée X) indique la position ou l'extrémité ouest de l'entité située le plus à l'ouest ? (La réponse désigne l'abscisse négative la plus petite à laquelle vous avez affaire.) Par exemple, si vous représentez des puits de pétrole et que certains soient situés à l'ouest du point d'origine, quelle abscisse indique la position du puits localisé le plus à l'ouest ?
- Existe-t-il des éléments situés au sud du point d'origine ? Si tel est le cas, quelle ordonnée (coordonnée Y) indique la position ou l'extrémité sud de l'entité située le plus au sud ? (La réponse désigne l'ordonnée négative la plus petite à laquelle vous avez affaire.) Par exemple, si vous représentez des puits de pétrole et que certains soient situés au sud du point d'origine, quelle ordonnée indique la position du puits localisé le plus au sud ?
- Si vous comptez utiliser des coordonnées Z pour définir la profondeur, quelle est l'entité située à la plus grande profondeur et quelle coordonnée Z représente le point le plus bas de cette entité ? (La réponse désigne la coordonnée Z négative la plus petite à laquelle vous avez affaire.)

- Si vous comptez inclure des mesures dans vos données spatiales, certaines seront-elles négatives ? Si tel est le cas, quelles est la plus petite d'entre elles ?

Après avoir vérifié les mesures et les coordonnées négatives les plus petites, ajoutez à chacune un montant égal à 5 % de sa valeur. Par exemple, si l'abscisse négative la plus petite est –100, vous pouvez ajouter –5. Dans ce manuel, le résultat de cette opération s'appelle une *valeur augmentée*.

Définition des facteurs de décalage

La prochaine étape consiste à spécifier les facteurs de décalage que DB2 Extension Spatiale doit appliquer pour convertir les coordonnées et mesures négatives en valeurs non négatives.

- Après avoir choisi la valeur d'abscisse (X) augmentée, spécifiez un décalage qui, lorsqu'il est soustrait de cette valeur, donne zéro. DB2 Extension Spatiale soustraira ensuite ce nombre de toutes les abscisses négatives afin d'obtenir une valeur positive. DB2 Extension Spatiale effectuera également la même opération sur les autres abscisses.

  Par exemple, si la valeur d'abscisse augmentée est –105, vous devez ôter –105 pour obtenir 0. DB2 Extension Spatiale soustraira ensuite –105 de toutes les abscisses des éléments que vous représentez. Aucune de ces coordonnées n'étant supérieure à –100, toutes les valeurs obtenues par cette opérations seront positives.
- De même, spécifiez des décalages qui, lorsque vous les soustrayez de la valeur d'ordonnée augmentée, de la valeur Z augmentée et de la mesure augmentée, donnent zéro.

Le décalage soustrait des abscisses s'appelle un *faux X*, ceux soustraits des ordonnées, des coordonnées Z et des mesures, respectivement *faux Y*, *faux Z* et *faux M*. Pour la procédure de définition de ces paramètres à partir du Centre de contrôle, reportez-vous à la section «Création d'un système de références spatiales à partir du Centre de contrôle» à la page 29.

Définition des facteurs d'échelle

La prochaine étape consiste à spécifier les facteurs d'échelle que DB2 Extension Spatiale doit appliquer pour convertir les coordonnées et mesures décimales en nombres entiers.

- Spécifiez un facteur d'échelle qui, multiplié par une abscisse ou une ordonnée décimale, donne un entier 32 bits. Il est recommandé d'opter pour une puissance de 10 : 10 à la puissance 1 (10), 10 à la puissance 2 (100), 10 à la puissance 3 (1000), ou à la puissance n, si besoin est. Pour choisir la puissance de 10 à utiliser :

  1. Identifiez les abscisses et les ordonnées qui sont des nombres décimaux ou sont susceptibles de l'être. Par exemple, supposons que, parmi les différentes abscisses et ordonnées que vous traitez, vous identifiez trois nombres décimaux : 1,23, 5,1235 et 6,789.

2. Prenez la coordonnée décimale avec la plus longue précision décimale. Déterminez ensuite quelle est la puissance de 10 par laquelle il faut multiplier cette coordonnée pour obtenir un entier doté d'une précision identique. Ainsi, parmi les trois coordonnées décimales de notre exemple, 5,1235 est celle qui a la plus longue précision. En la multipliant par 10 à la puissance 4 (10000), on obtient le nombre entier 51235.
3. Déterminez si l'entier obtenu par la multiplication décrite précédemment est trop long pour être stocké en tant qu'élément de données 32 bits. 51235 n'est pas trop long. Mais supposons qu'outre 1,23, 5,11235 et 6,789, votre plage d'abscisses et d'ordonnées comporte une quatrième valeur décimale : 10006,789876. La précision de ce nombre étant supérieure à celle des trois autres, vous devriez alors multiplier *cette* coordonnée, et non 5,1235, par une puissance de 10. Pour la convertir en nombre entier, il faudra la multiplier par 10 à la puissance 6 (1000000). Or, la valeur ainsi obtenue, soit 10006789876, est trop longue pour être stockée en tant qu'élément de données 32 bits. Par conséquent, si DB2 Extension Spatiale essaye de l'enregistrer, le résultat est imprévisible.

   Pour éviter ce type d'incident, optez pour une puissance de 10 qui, multipliée par la coordonnée d'origine, donne un nombre décimal que DB2 Extension Spatiale peut tronquer en un entier stockable, en perdant le minimum de précision. En l'occurrence, il peut s'agir de 10 à la puissance 4 (10000) ; 10000 multiplié par 10006,789876 égale 100067898,76. DB2 Extension Spatiale tronque ce nombre à 100067898, réduisant son exactitude de manière quasi-insignifiante.

- Si les entités représentées comportent des coordonnées Z décimales, suivez la procédure précédente pour vérifier le facteur d'échelle à appliquer à ces coordonnées. Si les entités sont associées à des mesures décimales, suivez la même procédure pour vérifier le facteur d'échelle à utiliser.

Le facteur d'échelle associé aux abscisses et aux ordonnées s'appelle *unité XY*, et ceux associés aux coordonnées Z et aux mesures, respectivement *unités Z* et *unités M*. Pour la procédure de définition de ces paramètres à partir du Centre de contrôle, reportez-vous à la section «Création d'un système de références spatiales à partir du Centre de contrôle».

Création d'un système de références spatiales à partir du Centre de contrôle

Cette section présente la procédure de création d'un système de références spatiales à partir du Centre de contrôle, puis décrit chaque étape de façon détaillée.

Ces étapes ne nécessitent aucun droit d'accès particulier.

Création d'un système de références spatiales à partir du Centre de contrôle - Présentation de la procédure

1. Ouvrez la fenêtre Création d'une référence spatiale.
2. Indiquez le système de coordonnées à utiliser.
3. Spécifiez les identificateurs du système de références spatiales que vous voulez créer.
4. Déterminez les plages de coordonnées et de mesures qui s'appliquent aux entités géographiques sur lesquelles vous désirez des informations.
5. Spécifiez les valeurs applicables pour convertir des coordonnées et mesures négatives ou décimales en éléments de données susceptibles d'être stockés par DB2 Extension Spatiale.
6. Indiquez à DB2 Extension Spatiale de créer le système de références spatiales de votre choix.

Création d'un système de références spatiales à partir du centre de contrôle - Description détaillée des étapes

1. Ouvrez la fenêtre Création d'une référence spatiale.
   a. Dans la fenêtre du Centre de contrôle, développez l'arborescence des objets jusqu'à ce que vous trouviez le dossier **Bases de données** sous le serveur sur lequel doit s'exécuter DB2 Extension Spatiale.
   b. Cliquez sur le dossier **Bases de données**. La liste des bases de données s'affiche dans le panneau de contenu figurant dans la partie droite de la fenêtre.
   c. Cliquez avec le bouton droit de la souris sur la base de données que vous souhaitez activer pour les opérations spatiales, puis sélectionnez **Extension Spatiale** —> **Références spatiales** dans le menu en incrustation. La fenêtre Références spatiales s'affiche.
   d. Dans la fenêtre Références spatiales, cliquez sur **Création**. La fenêtre Création d'une référence spatiale s'affiche.
2. Dans la fenêtre Création d'une référence spatiale, indiquez dans la zone **Système de coordonnées** le nom du système que vous souhaitez utiliser.
3. Spécifiez les identificateurs du système de références spatiales que vous voulez créer.
   - Dans la zone **Nom**, tapez un nom comportant de 1 à 64 caractères et désignant le système de références.

     Restriction : Ne spécifiez pas le nom d'un autre système de références spatiales. En effet, chaque nom de système de références spatiales doit être unique au sein d'une base de données.
   - Dans la zone ID, indiquez un identificateur numérique. Il doit s'agir d'un nombre entier.

     Restriction : Ne spécifiez pas l'ID d'un autre système de références spatiales. En effet, chaque ID de système de références spatiales doit être unique au sein d'une base de données.

4. En dehors du Centre de contrôle (sur une feuille de papier, un tableau blanc, etc.), déterminez les coordonnées et mesures négatives les plus petites associées aux entités géographiques que vous représentez. Pour ce faire, reportez-vous à la section «Détermination des coordonnées et mesures négatives les plus petites» à la page 27.
5. Dans la fenêtre Création d'une référence spatiale, spécifiez des valeurs permettant de convertir des coordonnées et mesures négatives ou décimales en éléments de données pris en charge par DB2 Extension Spatiale, c'est-à-dire des nombres entiers 32 bits non négatifs.
   a. Spécifiez des valeurs permettant de convertir des abscisses (coordonnées X) négatives ou décimales en nombres entiers non négatifs :
      - Dans la zone la plus proche du **X** de la colonne **Décalage**, spécifiez un faux X :
        - S'il existe des valeurs négatives dans la plage des abscisses identifiées à l'étape 4, tapez un faux X qui, soustrait de la coordonnée négative la plus petite, permet d'obtenir un nombre positif. Pour les instructions correspondantes, reportez-vous à la section «Définition des facteurs de décalage» à la page 28.
        - Si aucune des abscisses est négative, tapez un faux X ayant une valeur 0.
      - Dans la colonne **Facteur d'échelle**, spécifiez une unité XY dans la zone la plus éloignée à droite du **X**. Cette unité doit permettre, une fois multipliée par toute abscisse ou ordonnée décimale, d'obtenir un nombre complet qui puisse être stocké en tant qu'élément de donnée 32 bits, et ce, avec un minimum de déperdition au niveau de la précision. Pour les instructions correspondantes, reportez-vous à la section «Définition des facteurs d'échelle» à la page 28.

        Une fois l'unité XY spécifiée dans la zone la plus éloignée à droite du **X**, elle s'affiche également dans la zone la plus à droite du **Y**.
   b. Indiquez un faux Y qui permettra à DB2 Extension Spatiale de convertir les ordonnées en valeurs positives. Spécifiez-le dans la zone la plus proche du **Y** de la colonne **Décalage**.
      - S'il existe des valeurs négatives dans la plage des ordonnées identifiées à l'étape 4, tapez un faux Y qui, soustrait de la coordonnée négative la plus petite, permet d'obtenir un nombre positif. Pour les instructions correspondantes, reportez-vous à la section «Définition des facteurs de décalage» à la page 28.
      - Si toutes les ordonnées sont positives, tapez un faux Y ayant une valeur 0.
   c. Si vous comptez inclure des coordonnées Z dans vos données spatiales, spécifiez des valeurs permettant de convertir les coordonnées de ce type négatives ou décimales en nombres entiers non négatifs.

- Dans la zone la plus proche du **Z** de la colonne **Décalage**, spécifiez un faux Z :
  - S'il existe des valeurs négatives dans la plage des coordonnées Z identifiées à l'étape 4 à la page 31, tapez un faux Z qui, soustrait de la coordonnée négative la plus petite, permet d'obtenir un nombre positif. Pour les instructions correspondantes, reportez-vous à la section «Définition des facteurs de décalage» à la page 28.
  - Si aucune des coordonnées Z est négative, tapez un faux Z ayant une valeur 0.
- Dans la colonne **Facteur d'échelle**, spécifiez une unité Z dans la zone la plus éloignée à droite du **Z**. Cette unité doit permettre, une fois multipliée par toute coordonnée Z décimale, d'obtenir un nombre entier qui peut être stocké en tant qu'élément de donnée 32 bits, et ce, avec un minimum de déperdition au niveau de la précision. Pour les instructions correspondantes, reportez-vous à la section «Définition des facteurs d'échelle» à la page 28.

d. Si vous comptez inclure des mesures dans vos données spatiales, spécifiez des valeurs permettant de convertir les coordonnées de ce type négatives ou décimales en nombres entiers positifs.
   - Dans la colonne **Décalage**, spécifiez un faux M dans la zone la plus proche de l'intitulé **Linéaire** :
     - S'il existe des valeurs négatives dans la plage de mesures identifiées à l'étape 4 à la page 31, tapez un faux M qui, soustrait de la mesure négative la plus petite, permet d'obtenir un nombre positif. Pour les instructions correspondantes, reportez-vous à la section «Définition des facteurs de décalage» à la page 28.
     - Si toutes les mesures sont positives, tapez un faux M ayant une valeur 0.
   - Dans la colonne **Facteur d'échelle**, spécifiez une unité M dans la zone située le plus à droite de l'intitulé **Linéaire**. Cette unité doit permettre, une fois multipliée par toute mesure décimale, d'obtenir un nombre entier qui peut être stocké en tant qu'élément de donnée 32 bits, et ce, avec un minimum de déperdition au niveau de la précision. Pour les instructions correspondantes, reportez-vous à la section «Définition des facteurs d'échelle» à la page 28.

6. Cliquez sur **OK** pour créer le système de références spatiales de votre choix.

Chapitre 4. Définition et enregistrement des colonnes spatiales, et activation d'un géocodeur

Une fois les ressources de votre SIG DB2 Extension Spatiale configurées, la prochaine étape consiste à créer les objets qui contiendront les données spatiales. Par exemple, si vous avez besoin de nouvelles tables pour le stockage des données spatiales, vous pouvez les définir en affectant des types de données spatiales aux colonnes destinées à contenir les données. Si nécessaire, vous pouvez également ajouter des colonnes spatiales à des tables existantes.

Lorsque vous dotez une table, nouvellement créée ou existante, d'une colonne spatiale, vous devez enregistrer cette dernière en tant que couche. En outre, si vous envisagez d'utiliser un géocodeur pour peupler la colonne, vous pouvez, au moment de cet enregistrement, activer le géocodeur pour qu'il assure automatiquement la mise à jour de la colonne.Cette activation se produit ainsi : DB2 Extension Spatiale définit des déclencheurs qui sont codés pour appeler le géocodeur dès que la ou les colonnes d'attribut associées à la colonne spatiale reçoivent des données nouvelles ou mises à jour. Lorsqu'il reçoit l'appel, le géocodeur convertit les données nouvelles ou mises à jour en données spatiales et il place celles-ci dans la colonne spatiale.

Après avoir défini une colonne spatiale pour une table, vous pouvez, si vous le désirez, créer une colonne de vue sur cette colonne. Vous devez enregistrer la colonne de vue en tant que couche après avoir réalisé cette opération pour la colonne de la table.

Le présent chapitre décrit la nature et l'utilisation des types de données que vous pouvez affecter à une colonne spatiale. Il explique ensuite comment utiliser le Centre de contrôle pour définir une colonne spatiale destinée à une table, comment enregistrer cette colonne en tant que couche et activer un géocodeur pour la mettre à jour. Enfin, il indique comment enregistrer une colonne de vue en tant que couche à l'aide du Centre de contrôle.

Présentation des types de données spatiales

La présente section décrit les types de données requis pour les colonnes spatiales et vous indique comment choisir le type de données d'une colonne spatiale.

Lorsque vous activez une base de données pour des opérations spatiales, DB2 Extension Spatiale fournit à la base de données une hiérarchie de types de

données structurées. La figure 6, illustre cette hiérarchie. Dans cette figure, les types instanciables sont représentés sur fond blanc, les types non instanciables en grisé.

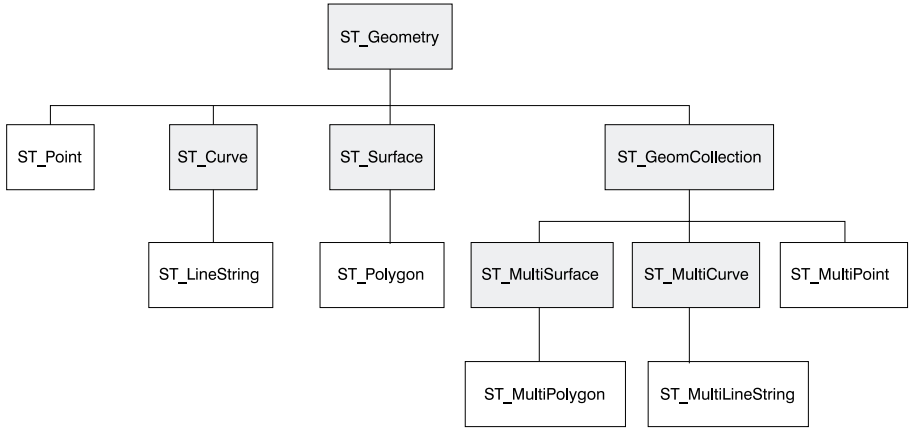


Figure 6. Hiérarchie des types de données spatiales. Les types de données figurant sur fond blanc sont instanciables et ceux représentés en grisé ne le sont pas.

La hiérarchie illustrée à la figure 6 comporte les types suivants :

- Types de données à associer à des entités géographiques pouvant être perçues comme étant composées d'une seule unité (habitations individuelles, lacs isolés, etc.).
- Types de données à associer à des entités géographiques constituées de plusieurs unités ou composants (systèmes autoroutiers, chaînes de montagnes, etc.).
- Type de données à associer à des entités géographiques diverses.

Types de données associés aux entités simples

Utilisez les types ST_Point, ST_Linestring et ST_Polygon pour stocker les coordonnées définissant l'espace occupé par des entités qui peuvent être perçues comme étant composées d'une seule unité.

- ST_Point permet d'indiquer la position d'une entité géographique discrète. Il peut s'agir d'une entité de très petite taille (puits, etc.), de très grande taille (ville, etc.) ou de taille moyenne (ensemble d'immeubles, parc, etc.). Dans chaque cas, la position peut être localisée à l'intersection d'un axe de coordonnées est-ouest (un parallèle, par exemple) et d'un axe de coordonnées nord-sud (un méridien, par exemple). Un élément de données ST_Point inclut des valeurs, une abscisse et une ordonnée, qui définissent ce type d'intersection. L'abscisse indique où se situe l'intersection sur l'axe est-ouest, et l'ordonnée, où elle se trouve sur l'axe nord-sud.
- ST_Linestring permet de stocker les coordonnées qui définissent l'espace occupé par des entités linéaires : rues, canaux, pipelines, etc.

- ST_Polygon permet d'indiquer la superficie occupée par une entité ayant plusieurs côtés : forêt, réserve naturelle, etc. Un élément de données ST_Polygon est constitué par les coordonnées qui définissent le périmètre de cette entité.

  Dans certains cas, vous pouvez utiliser à la fois ST_Polygon et ST_Point pour la même entité. Par exemple, supposons que vous ayez besoin d'informations spatiales sur différentes résidences. Si vous voulez représenter la position géographique de chaque résidence, vous devez utiliser le type ST_Point pour enregistrer les abscisses et les ordonnées définissant chaque position. Mais si vous voulez représenter la surface couverte par chaque résidence, vous devez recourir au type ST_Polygon pour stocker les coordonnées définissant le périmètre de chacune de ces zones.

Types de données associés aux entités composées

Utilisez les types ST_MultiPoint, ST_MultiLineString et ST_MultiPolygon pour stocker les coordonnées qui définissent la superficie occupée par des entités constituées de plusieurs unités :

- ST_MultiPoint permet de représenter des entités constituées d'unités discrètes et d'indiquer la position de chaque composant. Un élément de données ST_MultiPoint comprend les paires d'abscisse et d'ordonnée qui définissent la position de chaque composant de l'entité. Par exemple, prenons une table dont les lignes représentent des archipels et qui comporte, entre autres, une colonne ST_MultiPoint. Chaque élément de données de cette colonne comprend les couples d'abscisse et d'ordonnée qui définissent la position des îles dans chaque archipel.
- ST_MultiLineString permet de représenter des entités composées d'unités linéaires et d'avoir des informations sur l'espace occupé par chaque unité. Un élément de données ST_MultiLineString correspond aux coordonnées qui définissent de tels espaces. Par exemple, prenons une table dont les lignes représentent des systèmes hydrographiques et qui comporte, entre autres, une colonne ST_MultiLineString. Chaque élément de données de cette colonne comprend les ensembles de coordonnées qui définissent le tracé des rivières de chaque système.
- ST_MultiPolygon permet de représenter des entités constituées d'unités dotées de plusieurs côtés et d'avoir des informations sur la superficie occupée par chaque composant. Par exemple, prenons une table dont les lignes représentent les départements d'une région et qui comporte, entre autres, une colonne ST_MultiPolygon. Cette colonne contient des informations sur les zones agricoles. Plus précisément, chaque élément de données de la colonne contient les ensembles de coordonnées qui définissent le périmètre des zones agricoles dans un département déterminé.

Type de données pour entités de toutes sortes

Vous pouvez utiliser ST_Geometry lorsque vous avez un doute sur le type de données à associer. ST_Geometry figurant au sommet de la hiérarchie à laquelle appartiennent les autres types de données, une colonne ST_Geometry peut contenir tout ou partie des valeurs qui peuvent être stockées dans des colonnes associées aux autres types de données.

Attention : Le géocodeur par défaut peut convertir des adresses en éléments de données de type ST_Point ou ST_Geometry. Par conséquent, si vous envisagez de l'utiliser pour peupler une colonne spatiale, vous devez affecter à celle-ci l'un ces deux types.

Définition d'une colonne spatiale destinée à une table, enregistrement de cette colonne en tant que couche et activation du géocodeur pour la mettre à jour

Cette section présente les étapes permettant de définir une colonne spatiale destinée à une table, de l'enregistrer en tant que couche et d'activer le géocodeur pour la mettre à jour. Elle décrit ensuite chaque étape de façon détaillée.

Pour connaître les droits d'accès nécessaires à l'enregistrement d'une colonne de table en tant que couche, reportez-vous à la section «Classe d'autorisation» à la page 95, et pour ceux concernant l'activation d'un géocodeur à des fins de mise à jour de cette colonne, à la section «Classe d'autorisation» à la page 76.

Définition d'une colonne spatiale destinée à une table, enregistrement de cette colonne en tant que couche et activation du géocodeur pour la mettre à jour - Présentation de la procédure

1. Si la colonne spatiale est destinée à une nouvelle table, créez celle-ci.
2. Ouvrez la fenêtre Création d'une référence spatiale.
3. Ajoutez une colonne spatiale à une table et indiquez que vous voulez l'enregistrer en tant que couche, ou indiquez que vous voulez enregistrer une colonne existante en tant que couche.
4. Spécifiez le système de références spatiales à utiliser pour la couche concernée.
5. Si la couche contient des données importées ou générées à partir d'une autre colonne spatiale, demandez à DB2 Extension Spatiale de créer la couche.
6. Si la couche doit contenir des données dérivées de données d'attribut :
   a. Spécifiez la ou les colonnes qui contiennent ces données d'attribut.
   b. Précisez que vous voulez activer un géocodeur pour la mise à jour de la couche.

c. Demandez à DB2 Extension Spatiale de créer la couche.

Définition d'une colonne spatiale destinée à une table, enregistrement de cette colonne en tant que couche et activation du géocodeur pour la mettre à jour - Description détaillée des étapes

1. Si la colonne spatiale est destinée à une nouvelle table, créez celle-ci.
   - Utilisez l'interface de votre choix (Centre de contrôle, interpréteur de commandes, etc.) pour créer la table.
   - Si vous comptez utiliser un géocodeur, incluez jusqu'à dix colonnes sur lesquelles il exécutera des opérations. Un géocodeur ne peut pas utiliser plus de dix colonnes de données en entrée.
   - Insérez la colonne spatiale que vous enregistrerez en tant que couche ou définissez cette colonne à l'étape 3.

   Si vous comptez utiliser une table existante, passez à l'étape suivante.
2. Ouvrez la fenêtre Création d'une référence spatiale.
   a. Dans la fenêtre du Centre de contrôle, développez l'arborescence des objets jusqu'à ce que vous trouviez le dossier **Tables** contenant les tables de la base de donnée que vous utilisez pour les opérations spatiales.
   b. Cliquez sur le dossier **Tables**. La liste des tables s'affiche dans le panneau de contenu figurant dans la partie droite de la fenêtre.
   c. Cliquez avec le bouton droit de la souris sur la table de votre choix, puis sélectionnez **Extension Spatiale** —> **Couches spatiales** dans le menu en incrustation. La fenêtre Couches spatiales s'affiche.
   d. Dans la fenêtre Couches spatiales, cliquez sur **Création**. La fenêtre Création d'une couche spatiale s'affiche.
3. Dans la fenêtre Création d'une couche spatiale, ajoutez une colonne spatiale à une table et indiquez que vous voulez l'enregistrer en tant que couche, ou précisez que vous voulez enregistrer une colonne existante en tant que couche.
   - Si vous voulez ajouter une colonne spatiale à une table et la définir en tant que couche, procédez comme suit :
     a. Dans la zone **Colonne de couche**, tapez le nom de la colonne.
     b. Dans la zone **Type de colonne**, sélectionnez ou entrez le type de données à associer à la colonne. Pour connaître les types de données possibles, reportez-vous à la section «Présentation des types de données spatiales» à la page 33.
   - Si vous voulez définir une colonne existante en tant que couche, sélectionnez-la dans la zone **Colonne de couche**.

     Restriction : Ne sélectionnez pas une colonne déjà définie en tant que couche.

4. Dans la zone **Système de références spatiales**, indiquez le nom du système à associer à la couche.
5. Si la couche doit contenir des données importées ou générées à partir d'une autre colonne spatiale, cliquez sur **OK** pour l'enregistrer.
6. Si la couche doit contenir des données dérivées de données d'attribut, procédez comme suit :
   a. Spécifiez la ou les colonnes qui contiennent ces données d'attribut :
      1) Sélectionnez une ou plusieurs colonnes dans la boîte **Colonnes disponibles**. Vous êtes limité à 10.
      2) Cliquez sur l'un des boutons de fonction **>** ou **>>**, ou sur les deux pour transférer la ou les colonnes choisies dans la boîte **Colonnes sélectionnées**.
   b. Si vous voulez activer un géocodeur à des fins de mise à jour de la couche, procédez comme suit :
      1) Cochez la case **Géocodage automatique**.
      2) Dans la zone **Nom**, sélectionnez le nom du géocodeur que vous souhaitez utiliser.
      3) Dans la zone **Niveau de précision**, spécifiez sous forme de pourcentage le degré d'adéquation qui doit exister entre les enregistrements d'entrée et les enregistrements correspondants figurant dans le système de référence pour que les données d'entrée soient traitées. Ce pourcentage s'appelle une *précision*. Par exemple, supposons que le géocodeur lise un enregistrement d'entrée qui contienne l'adresse suivante : 557 Bailey, San Jose 94120. Si la précision est de 100 et que l'adéquation entre cette adresse et les données correspondantes figurant dans le système de référence n'est pas totale, autrement dit égale à 100 %, le géocodeur rejettera l'adresse. Si la précision est de 75 et que l'adéquation atteint au moins 75 %, le géocodeur la traitera.
      4) Si le géocodeur provient d'un fournisseur, la boîte **Propriétés** vous permet de spécifier tous les paramètres de géocodage du fournisseur, que vous souhaitez utiliser.
   c. Cliquez sur **OK** pour enregistrer la colonne sélectionnée en tant que couche et, si vous l'avez demandé, activer le géocodeur pour la mise à jour de la colonne.

Enregistrement d'une colonne de vue en tant que couche

Pour connaître les droits d'accès nécessaires à l'enregistrement d'une colonne de vue en tant que couche, reportez-vous à la section «Classe d'autorisation» à la page 95.

Pour enregistrer une colonne de vue en tant que couche, procédez comme suit :

1. Ouvrez la fenêtre Création d'une couche spatiale.
   a. Dans la fenêtre du Centre de contrôle, développez l'arborescence des objets jusqu'à ce que vous trouviez le dossier **Vues** contenant les vues de la base de donnée que vous utilisez pour les opérations spatiales.
   b. Cliquez sur le dossier **Vues**. La liste des vues s'affiche dans le panneau de contenu figurant dans la partie droite de la fenêtre.
   c. Cliquez avec le bouton droit de la souris sur la vue de votre choix, puis sélectionnez **Extension Spatiale** —> **Couches spatiales** dans le menu en incrustation. La fenêtre Couches spatiales s'affiche.
   d. Dans la fenêtre Couches spatiales, cliquez sur **Création**. La fenêtre Création d'une couche spatiale s'affiche.
2. Spécifiez le nom de la colonne à enregistrer en tant que couche dans la boîte **Colonne de couche**.
3. Dans la zone **Couche spatiale sous-jacente**, indiquez le nom de la colonne de table dont dépend la colonne de vue sélectionnée. Cette colonne de table doit être déjà enregistrée en tant que couche.
4. Cliquez sur **OK** pour enregistrer la colonne de vue spécifiée en tant que couche.

Chapitre 5. Peuplement des colonnes spatiales

Après avoir enregistré les colonnes spatiales en tant que couches, vous pouvez les alimenter en données spatiales. Comme indiquée à la section «Provenance des données spatiales» à la page 6, il existe trois méthodes pour obtenir ces données : utiliser une fonction appelée géocodeur pour les dériver de données d'attribut, utiliser d'autres fonctions pour les déduire d'autres données spatiales ou les importer de fichiers. Le présent chapitre traite des sujets suivants :

- géocodage et utilisation du Centre de contrôle pour géocoder les données d'attribut en traitement par lots ;
- importation et exportation de données, et utilisation du Centre de contrôle pour importer des données dans votre SIG et les en exporter.

Pour plus d'informations sur les fonctions permettant de dériver de nouvelles données spatiales à partir de données existantes du même type, reportez-vous à la section «Fonctions de génération de nouvelles géométries à partir de géométries existantes» à la page 156.

Géocodeurs

La présente section décrit le processus de géocodage et explique comment exécuter un géocodeur en traitement par lots à partir du Centre de contrôle.

Présentation du géocodage

La présente section explique les principales différences existant entre les géocodeurs et leurs sources de données. Elle décrit également les deux modes d'exécution possibles d'un géocodeur, ainsi que les facteurs à prendre en considération si vous envisagez d'utiliser ce type de fonction.

Avec DB2 Extension Spatiale, vous pouvez :

- Utiliser le géocodeur par défaut fourni avec DB2 Extension Spatiale.
- Télécharger des géocodeurs conçus par des fournisseurs non-IBM.
- Télécharger vos propres géocodeurs.

Le géocodeur par défaut convertit les adresses existant aux Etats-Unis en données spatiales de type ST_Point ou ST_Geometry. Si vous voulez stocker d'autres types de données spatiales, vous pouvez vous connecter à un géocodeur qui permet de les générer. Si vous avez besoin de données spatiales représentant des lieux situés hors des Etats-Unis ou sans adresse (zones agricoles, etc.), vous pouvez vous connecter à un géocodeur spécifique.

Pour qu'un géocodeur complémentaire puisse être utilisé, il doit avoir été préalablement enregistré par des utilisateurs ou des fournisseurs à l'aide de la procédure mémorisée db2gse.gse_register_gc. Il est impossible de l'enregistrer à partir du Centre de contrôle. Pour des informations spécifiques sur cette procédure, reportez-vous à la section «db2gse.gse_register_gc» à la page 93, et au «Chapitre 9. Procédures mémorisées» à la page 69, pour des informations d'ordre général sur l'utilisation des procédures mémorisées de DB2 Extension Spatiale.

Un géocodeur fonctionne selon deux modes différents :

- En *mode de traitement par lots*, il tente, en une seule opération, de convertir en données spatiales toutes les données source existantes destinées à une colonne spatiale et de peupler la colonne avec les données obtenues. Vous pouvez lancer ce processus à partir de la fenêtre Exécution d'un géocodeur ou au sein d'un programme d'application en codant celui-ci pour qu'il appelle la procédure mémorisée db2gse.gse_run_gc.
- En *mode incrémentiel*, un géocodeur convertit les données lorsqu'elles sont insérées ou mises à jour dans une table, et il place les valeurs spatiales obtenues dans une colonne afin que celle-ci soit constamment actualisée. Ce processus est activé par des déclencheurs d'insertion et de mise à jour que vous pouvez définir à partir de la fenêtre Création d'une couche spatiale, ou au sein d'un programme d'application en codant celui-ci pour qu'il appelle la procédure mémorisée db2gse.gse_enable_autogc.

  Le géocodage incrémentiel est également appelé *géocodage automatique*.

Lorsque vous envisagez d'utiliser un géocodeur, tenez compte des facteurs suivants :

1. Lorsque vous utilisez le Centre de contrôle, vous ouvrez normalement la fenêtre Création d'une couche spatiale avant la fenêtre Exécution d'un géocodeur. Cela signifie que, via DB2 Extension Spatiale, vous pouvez configurer des déclencheurs associés au géocodage incrémentiel avant d'avoir lancé le processus de géocodage en traitement par lots. Il est donc possible que le traitement incrémentiel intervienne avant le traitement par lots. Lors du traitement par lots de toutes les données source, le géocodeur est alors amené à traiter des données qu'il a déjà géocodé en mode incrémentiel. Si cette redondance n'entraîne pas de doublons (en effet, lorsque des données spatiales sont générées deux fois, les données obtenues la première fois sont remplacées par celles générées en dernier), elle peut cependant provoquer une dégradation des performances. Pour éviter ceci, vous pouvez différer la définition des déclencheurs afin qu'elle n'intervienne qu'à l'issue du processus de géocodage en traitement par lots.
2. Si les déclencheurs sont déjà définis lorsque vous vous apprêtez à géocoder les données source en traitement par lots, nous vous recommandons de les désactiver jusqu'à ce que cette opération soit

terminée. Vous pouvez le faire à partir de la fenêtre Exécution d'un géocodeur ou d'un programme d'application en le codant pour qu'il appelle la procédure mémorisée db2gse.gse_disable_autogc. Dans le premier cas, DB2 Extension Spatiale réactive automatiquement les déclencheurs à l'issue du géocodage, dans le second, vous pouvez les réactiver en appelant la procédure mémorisée db2gse.gse_enable_autogc.

3. Si vous souhaitez exécuter un géocodeur en traitement par lots pour peupler une colonne spatiale dotée d'un index, désactivez ou supprimez d'abord l'index. En effet, si l'index est toujours opérationnel pendant l'exécution du géocodeur, on constate une importante dégradation des performances. Si vous utilisez le Centre de contrôle, vous pouvez désactiver l'index à partir de la fenêtre Exécution d'un géocodeur. DB2 Extension Spatiale réactive automatiquement l'index à l'issue du géocodage. Si vous recourez à un programme d'application, vous pouvez supprimer l'index à l'aide de l'instruction SQL DROP. Dans ce cas, veillez à noter les paramètres de l'index afin de pouvoir recréer celui-ci une fois le géocodage en traitement par lots achevé.
4. Lorsque le géocodeur lit un enregistrement de données source, il tente de trouver un enregistrement correspondant dans les données de référence. Cette correspondance doit atteindre un certain degré d'exactitude (appelé *précision*) pour que le géocodeur traite l'enregistrement. Ainsi, une précision de 85 signifie que la correspondance entre l'enregistrement source et les données de référence doit être d'au moins 85 % pour que l'enregistrement source soit traité.

   Vous spécifiez la précision voulue, mais vous serez éventuellement amené à la modifier. Par exemple, supposons que la précision soit de 100. Si de nombreux enregistrements source contiennent des adresses plus récentes que les données de référence, il sera impossible d'obtenir un niveau de correspondance de 100 % entre ces enregistrements et les données de référence. Par conséquent, le géocodeur rejettera les enregistrements source. En règle générale, si un géocodeur génère des données spatiales qui paraissent insuffisantes ou très imprécises, vous parviendrez éventuellement à résoudre ce problème en modifiant la précision et en relançant le géocodeur.

Exécution du géocodeur en traitement par lots

Cette section présente la procédure d'exécution d'un géocodeur en traitement par lots à partir du Centre de contrôle. Elle décrit ensuite chaque étape de façon détaillée.

Pour connaître les droits d'accès nécessaires à l'exécution d'un géocodeur dans ce mode, reportez-vous à la section «Classe d'autorisation» à la page 102.

Exécution d'un géocodeur en traitement par lots - Présentation de la procédure

1. Ouvrez la fenêtre Exécution d'un géocodeur.
2. Indiquez le nom du géocodeur à utiliser.
3. Désactivez les objets susceptibles de dégrader les performances du géocodeur.
4. Spécifiez le nombre d'enregistrements à géocoder avant que DB2 n'émette une demande de validation.
5. Indiquez le mode d'exécution du géocodeur.
6. Demandez à DB2 Extension Spatiale d'exécuter le géocodeur.

Exécution d'un géocodeur en traitement par lots - Description détaillée des étapes

1. Ouvrez la fenêtre Exécution d'un géocodeur.
   a. Dans la fenêtre du Centre de contrôle, développez l'arborescence des objets jusqu'à ce que vous trouviez le dossier **Tables** de la base de donnée que vous utilisez pour les opérations spatiales.
   b. Cliquez sur le dossier **Tables**. La liste des tables s'affiche dans le panneau de contenu figurant dans la partie droite de la fenêtre.
   c. Cliquez avec le bouton droit de la souris sur la table de votre choix, puis sélectionnez **Couches spatiales** dans le menu en incrustation. La fenêtre correspondante s'affiche.
   d. Dans la fenêtre Couches spatiales :
      1) Sélectionnez la couche définie sur la colonne que vous voulez peupler.
      2) Cliquez sur le bouton de fonction **Exécution d'un géocodeur**. Ouvrez la fenêtre Exécution d'un géocodeur.
2. Si vous voulez utiliser le géocodeur par défaut, ne modifiez pas le contenu la boîte **Nom** qui affiche la valeur par défaut. Sinon, sélectionnez le géocodeur de votre choix dans cette zone.
3. Désactivez les objets susceptibles de dégrader les performances du géocodeur :
   - Si la colonne à peupler est dotée d'un index, cochez la case **Désactivation temporaire des index spatiaux pour la durée du géocodage**.
   - Si des déclencheurs ont été définis pour activer le géocodage incrémentiel de cette colonne, cochez la case **Désactivation temporaire des déclencheurs spatiaux pour la durée du géocodage**.

     Les index et les déclencheurs sont automatiquement réactivés lorsque vous cliquez sur **OK** dans la fenêtre Exécution d'un géocodeur.
4. Spécifiez le nombre d'enregistrements à géocoder avant que DB2 n'émette une demande de validation à l'aide du sélecteur **Portée de validation**. Par exemple, pour que DB2 valide en une fois un lot de 100 enregistrements géocodés, spécifiez le nombre 100.

Suggestion : Pour que DB2 ne lance la validation qu'une fois tous les enregistrements traités, spécifiez zéro.

5. Les zones de la boîte d'options **Paramètres du géocodeur** vous permettent d'indiquer le mode de fonctionnement du géocodeur :
   - Le sélecteur **Précision** permet de spécifier, sous forme de pourcentage, le degré d'exactitude qui doit exister entre les enregistrements source et les données de référence correspondantes. Pour plus d'informations sur ce paramètre, reportez-vous à la section «Présentation du géocodage» à la page 41.
   - Si vous utilisez un géocodeur provenant d'un fournisseur et que vous souhaitez tirer parti de propriétés qu'il prend en charge, définissez celles-ci dans la boîte **Propriétés**.
   - Si vous ne voulez géocoder qu'un sous-ensemble de lignes de la table que vous avez sélectionnée, utilisez la boîte **Clause WHERE** pour coder une clause SELECT WHERE qui précisera les critères à appliquer aux lignes en question. Cette clause peut faire référence à toutes les lignes de la table.

     Ne tapez que les critères sans le mot clé WHERE. Par exemple, si la table comporte une colonne STATE et que vous ne voulez géocoder que les lignes contenant la valeur MA dans cette colonne, tapez :

     ```
     STATE='MA'
     ```
6. Cliquez sur **OK** pour exécuter le géocodeur.

Importation et exportation de données

La présente section décrit les processus d'importation et d'exportation de données, et explique comment utiliser le Centre de contrôle pour effectuer les opérations suivantes :

- Importation de données provenant d'un fichier d'échange de données dans une table nouvellement créée ou existante
- Importation de données provenant d'un fichier d'échange de données dans une table existante
- Exportation de données provenant d'une table dans un fichier d'échange de données

Importation et exportation

La présente section répertorie des arguments justifiant l'importation et à l'exportation de données spatiales. Elle traite également des fichiers d'échange de données qui servent d'interface entre les sources de l'exportation et les cibles de l'importation.

DB2 Extension Spatiale permet d'importer des données spatiales provenant de fichiers d'échange de données ou d'en exporter vers ce type de fichiers. Considérons les scénarios suivants :

- Votre SIG contient des données spatiales qui représentent vos bureaux, vos clients et d'autres éléments de votre activité. Vous voulez y ajouter des données spatiales qui représentent l'environnement culturel de votre entreprise (villes, rues, sites présentant un intérêt, etc.). Or ces données sont disponibles auprès d'un fournisseur de cartes. DB2 Extension Spatiale vous permet de les importer à partir d'un fichier d'échange de données fourni par cette société.
- Vous voulez faire migrer des données spatiales d'un système Oracle vers votre système SIG DB2 Extension Spatiale. Pour ce faire, vous recourez à un utilitaire Oracle qui permet de charger les données dans un fichier d'échange. Ensuite, vous utilisez DB2 Extension Spatiale pour importer les données de ce fichier dans la base de données que vous avez activée pour les opérations spatiales.
- Vous souhaitez utiliser un navigateur SIG pour présenter des informations spatiales à vos clients sous une forme visuelle. Le navigateur n'a besoin que de fichiers, il n'est pas nécessaire qu'il soit connecté à une base de données. Vous pouvez utiliser DB2 Extension Spatiale pour exporter les données dans un fichier d'échange de données, puis un utilitaire du navigateur pour les charger dans ce dernier.

Le Centre de contrôle prend en charge deux types de fichiers d'échange de données pour DB2 Extension Spatiale : les fichiers de formes (SHAPE) et les fichiers de transfert ESRI_SDE. Les fichiers SHAPE sont souvent utilisés pour l'importation de données provenant de systèmes de fichiers et pour l'exportation de données vers des fichiers devant être chargés dans des systèmes de fichiers. Les fichiers de transfert ESRI_SDE servent fréquemment à l'importation de données provenant de bases de données ESRI.

Importation de données dans une table nouvellement créée ou existante

Cette section présente la procédure d'importation de données provenant d'un fichier SHAPE ou de transfert ESRI_SDE dans une table existante ou nouvellement créée. Elle décrit ensuite chaque étape de façon détaillée.

Afin de connaître les droits requis pour l'importation de formes, reportez-vous à la section «Classe d'autorisation» à la page 91, et à la section «Classe d'autorisation» à la page 89, pour ceux liés à l'importation de données ESRI_SDE.

Importation de données dans une table nouvellement créée ou existante - Présentation de la procédure

1. Ouvrez la fenêtre Importation de données spatiales.

2. Spécifiez le chemin d'accès, le nom et le format du fichier contenant les données à importer.
3. Spécifiez le nombre d'enregistrements à importer avant chaque validation.
4. En cas d'importation des données spatiales dans une table qui doit être créée, indiquez le nom de cette table et le nom de la colonne qui doit accueillir les données. S'il s'agit d'importer des données spatiales dans une table qui existe déjà, précisez le nom de la colonne destinée à contenir les données.
5. Indiquez le système de références spatiales à associer aux données.
6. Désignez un fichier destiné à recevoir les enregistrements dont l'importation échoue.
7. Indiquez à DB2 Extension Spatiale d'importer les données et, si vous avez défini une table dans cette fenêtre, de créer la table et d'enregistrer en tant que couche la colonne qui doit recevoir les données.

Importation de données dans une table nouvellement créée ou existante - Description détaillée des étapes

1. Ouvrez la fenêtre Importation de données spatiales.
   a. Dans la fenêtre du Centre de contrôle, développez l'arborescence des objets jusqu'à ce que vous trouviez le dossier **Bases de données** sous le serveur sur lequel doit s'exécuter DB2 Extension Spatiale.
   b. Cliquez sur le dossier **Bases de données**. La liste des bases de données s'affiche dans le panneau de contenu figurant dans la partie droite de la fenêtre.
   c. Cliquez avec le bouton droit de la souris sur la base de données dans laquelle vous voulez importer des données, puis cliquez sur **Extension Spatiale —> Importation de données spatiales** dans le menu en incrustation. La fenêtre Importation de données spatiales s'ouvre.
2. Spécifiez le chemin d'accès, le nom et le format du fichier contenant les données à importer :
   a. La zone **Nom de fichier** permet de spécifier le chemin d'accès et le nom du fichier.
   b. La zone **Format de fichier** de préciser le format. Vous avez le choix entre :

      Shape Il s'agit de la valeur par défaut.

      ESRI_SDE
      Si vous spécifiez ce format, la zone **Système de références spatiales** prend comme valeur par défaut le nom du système de références spatiales associé à ce format.

3. La zone **Portée de validation** permet de spécifier le nombre d'enregistrements qui doivent être importés avant chaque validation. Par exemple, pour que DB2 valide en une fois un lot de 100 enregistrements, spécifiez le nombre 100.

   Suggestion : Pour que DB2 ne lance la validation qu'une fois tous les enregistrements traités, spécifiez zéro.
4. Spécifiez la table et la colonne qui doivent recevoir les données.
   a. Dans la boîte **Schéma de couche**, indiquez le schéma à associer à la table destinée à recevoir les données importées.
   b. Spécifiez le nom de la table et de la colonne :
      - Si la table n'est pas encore créée :
        1) Dans la zone **Table de couche**, tapez un nom de table.
        2) Dans la zone **Colonne de couche**, tapez le nom de la colonne qui doit contenir les données importées. DB2 Extension Spatiale enregistre automatiquement cette colonne en tant que couche.
      - Si la table existe déjà :
        1) Dans la zone **Table de couche**, spécifiez le nom de la table. Elle doit déjà contenir la colonne destinée à recevoir les données importées. En outre, cette colonne doit déjà être enregistrée en tant que couche.
        2) Dans la zone **Colonne de couche**, indiquez le nom de la colonne qui doit contenir les données importées.
5. Dans la zone **Système de références spatiales**, tapez ou sélectionnez le nom du système à associer à ces données. Si les données doivent provenir d'un fichier de transfert ESRI_SDE, le nom du système de références spatiales associé s'affiche automatiquement dans la zone.
6. Dans la zone **Fichier d'exceptions**, spécifiez le chemin d'accès et le nom du nouveau fichier destiné à recevoir les enregistrements dont l'importation échoue. Vous pourrez ultérieurement corriger ces enregistrements et les importer à partir de ce fichier.

   DB2 Extension Spatiale crée ce fichier, ne spécifiez donc pas le nom d'un fichier qui existe déjà.
7. Cliquez sur **OK** pour importer les données. Par ailleurs, si vous avez indiqué le nom d'une table qui n'existe pas encore, celle-ci sera alors créée et la colonne destinée à recevoir les données sera enregistrée en tant que couche. Enfin, le fichier d'exceptions que vous avez spécifié sera également généré à ce stade.

Importation de données dans une table existante

Cette section présente la procédure d'importation de données provenant d'un fichier SHAPE ou de transfert ESRI_SDE dans une table existante. Elle décrit ensuite chaque étape de façon détaillée.

Afin de connaître les droits requis pour l'importation de formes, reportez-vous à la section «Classe d'autorisation» à la page 91, et à la section «Classe d'autorisation» à la page 89, pour ceux liés à l'importation de données ESRI_SDE.

Importation de données dans une table existante - Présentation de la procédure

1. Ouvrez la fenêtre Importation de données spatiales.
2. Spécifiez le chemin d'accès et le nom du fichier contenant les données à importer.
3. Spécifiez le nombre d'enregistrements à importer avant chaque validation.
4. Spécifiez la colonne destinée à contenir les données spatiales que vous importez.
5. Indiquez le système de références spatiales à associer aux données.
6. Désignez un fichier destiné à recevoir les enregistrements dont l'importation échoue.
7. Indiquez à DB2 Extension Spatiale d'importer les données et, si vous avez spécifié une colonne qui n'est pas encore créée, de la créer et de l'enregistrer en tant que couche.

Importation de données dans une table existante - Description détaillée des étapes

1. Ouvrez la fenêtre Importation de données spatiales.
   a. Dans la fenêtre du Centre de contrôle, développez l'arborescence des objets jusqu'à ce que vous trouviez le dossier **Tables** de la base de donnée que vous utilisez pour les opérations spatiales.
   b. Cliquez sur le dossier **Tables**. La liste des tables s'affiche dans le panneau de contenu figurant dans la partie droite de la fenêtre.
   c. Cliquez avec le bouton droit de la souris sur la base de données dans laquelle vous importez des données, puis cliquez sur **Extension Spatiale** —> **Importation de données spatiales** dans le menu en incrustation. La fenêtre Importation de données spatiales s'ouvre.
2. Dans la zone **Nom de fichier**, spécifiez le chemin d'accès et le nom du fichier contenant les données à importer.
3. La zone **Portée de validation** permet de spécifier le nombre d'enregistrements qui doivent être importés avant chaque validation. Par exemple, pour que DB2 valide un lot de 100 enregistrements, spécifiez le nombre 100.

   Suggestion : Pour que DB2 ne lance la validation qu'une fois tous les enregistrements traités, spécifiez zéro.
4. Spécifiez la colonne destinée à contenir les données spatiales que vous importez.

- Si la colonne n'existe pas encore, tapez le nom de votre choix dans la zone **Colonne de couche**.
- Si la colonne existe déjà, sélectionnez son nom ou tapez-le dans la zone **Colonne de couche**.

5. Dans la zone **Système de références spatiales**, indiquez le nom du système à associer à ces données.
   - Si vous ajoutez une colonne à une table, tapez ou sélectionnez le nom du système de références spatiales.
   - Si les données importées sont destinées à une colonne existante, ne modifiez pas la zone **Système de références spatiales**. En effet, elle affiche le nom du système de références spatiales par défaut.
6. Dans la zone **Fichier d'exceptions**, spécifiez le chemin d'accès et le nom du nouveau fichier destiné à recevoir les enregistrements dont l'importation échoue. Vous pourrez ultérieurement corriger ces enregistrements et les importer à partir de ce fichier.

   DB2 Extension Spatiale génère ce fichier, ne spécifiez donc pas le nom d'un fichier qui existe déjà.
7. Cliquez sur **OK** pour importer les données. Par ailleurs, si vous avez indiqué le nom d'une colonne qui n'existe pas encore, celle-ci sera alors créée et enregistrée en tant que couche. Enfin, le fichier d'exceptions que vous avez spécifié sera également créé.

Exportation de données vers un fichier de formes

Cette section présente la procédure d'exportation de données vers un fichier SHAPE. Elle décrit ensuite chaque étape de façon détaillée.

Afin de connaître les droits requis pour l'exécution de ces étapes, reportez-vous à la section «Classe d'autorisation» à la page 87.

Exportation de données vers un fichier de formes - Présentation de la procédure

1. Ouvrez la fenêtre Exportation de données spatiales.
2. Spécifiez la colonne qui contient les données spatiales à exporter.
3. Si vous souhaitez exporter un sous-ensemble de lignes de données, identifiez ce sous-ensemble pour DB2 Extension Spatiale.
4. Spécifiez le chemin d'accès et le nom du fichier vers lequel exporter les données.
5. Indiquez à DB2 Extension Spatiale d'exporter les données.

Exportation de données vers un fichier de formes - Description détaillée des étapes

1. Ouvrez la fenêtre Exportation de données spatiales.
   a. Dans la fenêtre du Centre de contrôle, développez l'arborescence des objets jusqu'à ce que vous trouviez le dossier **Tables** ou **Vues** de la base de donnée contenant les données spatiales.
   b. Cliquez sur le dossier **Tables** ou **Vues**. La liste des tables ou des vues s'affiche dans le panneau de contenu figurant dans la partie droite de la fenêtre.
   c. Cliquez avec le bouton droit de la souris sur la table ou la vue contenant les données à exporter, puis cliquez sur **Extension Spatiale —> Exportation de données spatiales** dans le menu en incrustation. La fenêtre Exportation de données spatiales s'ouvre.
2. Dans la zone **Colonne de couche**, indiquez le nom de la colonne qui contient les données à exporter.
3. Si vous voulez exporter un sous-ensemble de lignes de la table, utilisez la boîte **Clause WHERE** pour taper une clause de ce type qui spécifie les critères à appliquer aux lignes en question. Dans cette clause, vous ne pouvez faire référence qu'à des colonnes de la table ou de la vue à partir de laquelle vous exportez des données.

   Ne tapez que les critères sans le mot clé WHERE. Par exemple, si la table ou la vue comporte une colonne STATE et que vous ne voulez géocoder que les lignes contenant la valeur MA dans cette colonne, tapez :

   ```
   STATE='MA'
   ```
4. Dans la zone **Nom de fichier**, spécifiez le chemin d'accès et le nom du fichier dans lequel vous voulez exporter les données.
5. Cliquez sur **OK** pour exporter les données.

Chapitre 6. Création d'index spatiaux

Le présent chapitre explique comment créer un index associé à vos données spatiales à l'aide du Centre de contrôle.

Après avoir peuplé les colonnes spatiales avec vos données, vous pouvez créer un index spatial. Des structures d'indexation classiques (arbre B, etc.) exécutent des tris linéaires, unidimensionnels sur des données de type table. Des données de ce type qui ont été activées pour les opérations spatiales ne sont pas stockées en tant qu'entrée unique, elles sont bidimensionnelles. Par exemple, les formes géométriques spatiales telles qu'un polygone consistent en plusieurs coordonnées contenues dans une colonne ou couche spatiale. Un index en arbre B ne pouvant pas traiter les types de données spatiales, DB2 Extension Spatiale a donc créé un système d'indexation propriétaire appelé *index de grille*. Cet index est dérivé de l'index en arbre B, qui a été amélioré pour traiter les données bi-dimensionnelles et exécuter une indexation sur des colonnes spatiales. L'index de grille prend trois couches en charge et il est conçu pour donner de bonnes performances avec un large éventail d'objets, de trames et de distributions de données. Pour plus d'informations sur les index spatiaux, reportez-vous au «Chapitre 12. Index spatiaux» à la page 119.

Afin de connaître les droits requis pour la création d'un index spatial, reportez-vous à la section «Classe d'autorisation» à la page 81.

Création d'un index spatial à l'aide du Centre de contrôle

Pour créer un index spatial à partir du Centre de contrôle, procédez comme suit :

1. Dans l'arborescence des objets, sélectionnez le dossier **Tables**. La liste des tables existantes s'affiche dans le panneau de contenu.
2. Dans ce panneau, cliquez avec le bouton droit de la souris sur la table pour laquelle vous voulez créer un index, puis cliquez sur **Extension Spatiale** —> **Index spatiaux** dans le menu en incrustation. La fenêtre correspondante s'affiche.
3. Dans la fenêtre Index spatiaux, cliquez sur **Création**. La fenêtre Création d'un index spatial s'ouvre.
4. Dans la zone **Nom**, tapez le nom de l'index spatial que vous souhaitez créer.

   Remarque : Vous n'avez pas besoin de spécifier un schéma. En effet, DB2 Extension Spatiale ajoute automatiquement le schéma et crée un nom qualifié complet à votre place.

5. Dans la zone **Colonne de couche**, sélectionnez la couche pour laquelle vous créez un index.

   Une couche est une colonne spatiale définie ou enregistrée dans DB2 Extension Spatiale.

6. Dans les zones **Trame de la grille**, indiquez la valeur de trame à affecter à chaque zone.

   Les niveaux de trame, **Serrée**, **Moyenne** et **Large**, sont spécifiés en augmentant la taille de la cellule. Ainsi, le niveau intermédiaire doit avoir une cellule d'une taille supérieure au premier niveau, et le troisième, une cellule d'une taille supérieure au second.

Détermination de la taille des cellules de la grille

La détermination de la taille convenant à la grille s'effectue par un processus d'essais et d'erreurs. Nous vous recommandons de définir la trame de la grille en fonction de la taille approximative de l'objet indexé. Les trames de grille trop serrées ou trop larges peuvent entraîner un ralentissement des performances du système. Un maillage trop serré affecte le rapport clé/objet pendant une recherche par index. Dans ce cas, un trop grand nombre de clés est créé et trop de candidats sont renvoyés. Avec un maillage trop large, la recherche par index initiale renvoie un petit nombre de candidats, mais les performances risquent d'être ralenties pendant le balayage final de la table.

Pour plus d'informations sur la sélection de la taille des cellules d'une grille et le nombre de niveaux de trame, reportez-vous à la section «Sélection de la taille des cellules de la grille» à la page 126.

Chapitre 7. Extraction et analyse d'informations spatiales

Une fois les index spatiaux créés, les tables spatiales sont opérationnelles. Ce chapitre traite des sujets relatifs à l'extraction et à l'analyse de données spatiales. Il présente les différentes méthodes d'extraction et fournit des exemples de requêtes portant sur des tables et utilisant des fonctions spatiales.

Méthodes d'analyse des données spatiales

Vous pouvez analyser des données spatiales à l'aide de fonctions SQL et spatiales dans tous les environnements de programmation suivants :

- Navigateur géographique (par exemple, ArcExplorer développé par l'ESRI).

  Pour plus d'informations sur l'installation et l'utilisation du logiciel ArcExplorer, reportez-vous au manuel *Using ArcExplorer*, disponible sur le site Web ESRI à l'adresse http://www.esri.com.
- Instructions SQL interactives.
- Applications utilisateur (par exemple, ODBC, JDBC et SQL imbriqué).

Applications exécutables à partir du Centre de contrôle DB2, de la fenêtre de commande DB2 ou de l'interpréteur de commandes.

Création d'une requête spatiale

La présente section explique comment créer des requêtes spatiales qui utilisent des fonctions et prédicats spatiaux.

Fonctions spatiales et SQL

DB2 Extension Spatiale comprend des fonctions permettant d'effectuer diverses opérations sur les données spatiales. Les exemples décrits dans la présente section vous indiquent comment utiliser des fonctions spatiales pour créer vos propres requêtes spatiales.

Le tableau 3 contient une liste des fonctions spatiales et des opérations correspondantes.

Tableau 3. Fonctions spatiales et opérations

Type de fonction	Exemple d'opération
Calcul	Calcul de la distance entre deux points
Comparaison	Trouver tous les clients situés dans une zone d'inondation
Echange de données	Conversion des données en formats pris en charge
Transformation	Ajouter un rayon de 7 km autour d'un point

Pour plus d'informations sur les fonctions spatiales, reportez-vous au «Chapitre 13. Géométries et fonctions spatiales associées» à la page 129, et au «Chapitre 14. Fonctions spatiales associées aux requêtes SQL» à la page 167.

Exemple 1 : Comparaison

La requête ci-après permet d'identifier la distance moyenne existant entre les clients et chaque grand magasin. Les fonctions spatiales utilisées sont les suivantes : ST_Distance et 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
```

Exemple 2 : Echange de données

La requête ci-après permet de localiser les clients qui habitent dans la région de San Francisco. Les fonctions spatiales utilisées sont les suivantes : ST_AsText (échange de données) et ST_Within. ST_AsText convertit les données spatiales de la colonne c.location au format OGC TEXT.

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

Exemple 3 : Calcul

La requête ci-après permet de trouver toutes les rues dépassant 16 km de longueur. La fonction spatiale utilisée est la suivante : ST_Length.

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

Exemple 4 : Transformation

La requête ci-après permet d'identifier les clients qui vivent à l'intérieur de la zone d'inondation ou dans un périmètre de 3 km au-delà de cette zone. Les fonctions spatiales utilisées sont les suivantes : ST_Buffer (transformation) et ST_Within.

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

Prédicats spatiaux et SQL

Une catégorie particulière de fonctions spatiales appelées prédicats permet d'améliorer les performances des requêtes. Les prédicats spatiaux, tels ST_Overlaps qui compare deux polygones pour voir s'ils se chevauchent, peuvent s'avérer coûteux en termes de temps et de mémoire. Par conséquent, les techniques d'optimisation permettant de minimiser les coûts d'exécution revêtent une grande importance. L'optimiseur de requêtes DB2 utilise les index spatiaux pour améliorer les performances des requêtes lorsque vous employez les prédicats spatiaux en respectant les règles décrites

ultérieurement dans cette section. Pour plus d'informations sur les prédicats spatiaux, reportez-vous à la section «Prédicats» à la page 143. Les prédicats spatiaux utilisés pour l'exploitation des index spatiaux sont les suivants :

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

Pour consulter la liste exhaustive des fonctions et prédicats spatiaux, reportez-vous au «Chapitre 14. Fonctions spatiales associées aux requêtes SQL» à la page 167.

Règles d'exploitation des index

Respectez les règles suivantes si vous voulez optimiser des requêtes spatiales à l'aide de prédicats spatiaux.

- Le prédicat doit figurer dans la clause WHERE.
- Le prédicat doit figurer à gauche de la comparaison. Par exemple :

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

- Les comparaisons d'égalité doivent utiliser la constante entière 1.

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

- Dans le prédicat, une colonne spatiale doit être utilisée en tant que cible de la recherche et elle doit être dotée d'un index spatial.

Exemples d'exploitation d'index

Le tableau 4 indique les manières correcte et incorrecte de créer des requêtes spatiales dans le cadre de l'exploitation de l'index spatial.

Tableau 4. Règles d'exploitation des index

Requête spatiale	Règle non respectée

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Within(c.location,:BayArea)=1`
	Toutes les conditions sont respectées dans cet exemple.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Distance(c.location,:SanJose)<10`
	Toutes les conditions sont respectées dans cet exemple.

Tableau 4. Règles d'exploitation des index (suite)

Requête spatiale	Règle non respectée

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Length(c.location)>10`
	Le prédicat doit figurer dans la clause WHERE. (ST_Length est une fonction spatiale mais *pas* un prédicat.)

`SELECT *`
`FROM customers c`
`WHERE 1=db2gse.ST_Within(c.location,:BayArea)`
	Le prédicat doit figurer à gauche de la comparaison.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Within(c.location,:BayArea)=2`
	Les comparaisons d'égalité doivent utiliser la constante entière 1.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Within(:SanJose,:BayArea)=1`
	Dans le prédicat, une colonne spatiale doit être utilisée en tant que cible de la recherche et elle doit être dotée d'un index spatial. (SanJose et BayArea ne sont pas des colonnes spatiales et ne peuvent donc pas être associées à un index spatial.)

Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale

Le présent chapitre explique comment utiliser le programme exemple DB2 Extension Spatiale pour rédiger des applications permettant d'utiliser et de personnaliser des informations spatiales. Il aborde les sujets suivants :

- Utilisation du programme exemple
- Etapes du programme exemple

Utilisation du programme exemple

Le programme exemple DB2 Extension Spatiale simplifie la programmation d'applications. Il permet de :

- Automatiser des procédures spatiales récurrentes
- Intégrer par copier-coller du code exemple dans vos propres applications
- Comprendre les étapes généralement nécessaires pour créer et mettre à jour une base de données activée pour les opérations spatiales.

Le programme exemple permet de coder des tâches complexes pour DB2 Extension Spatiale ; par exemple, rédiger une application qui utilise l'interface de la base de données pour appeler des procédures mémorisées DB2 Extension Spatiale. A partir de ce programme, vous pouvez copier et personnaliser vos applications. Si vous n'êtes pas familiarisé avec la procédure de programmation de DB2 Extension Spatiale, vous pouvez exécuter le programme exemple qui en détaille chaque étape. Toutefois, vous devez tout d'abord créer le programme exemple. Pour ce faire, vous devez utiliser le fichier makefile. Pour la procédure de création et d'exécution du programme exemple, reportez-vous à la section «Vérification de l'installation» à la page 20.

Etapes du programme exemple

Le tableau 5 à la page 60, répertorie les étapes constituant le programme exemple, ainsi que les procédures mémorisées associées, et comporte une description de chaque étape. Les fonctions C permettant d'appeler les procédures mémorisées figurent entre parenthèses dans la colonne Action. Pour plus d'informations sur les procédures mémorisées, reportez-vous au «Chapitre 9. Procédures mémorisées» à la page 69. Le programme exemple est bâti sur le scénario décrit à la section «Scénario : mise à jour du SIG d'une compagnie d'assurance» à la page 13.

Tableau 5. Programme exemple DB2 Extension Spatiale

Etapes du programme exemple	Action	Description
Activation/ désactivation de la base de données spatiales	1. Activation de la base de données spatiale (gseEnableDB) 2. Désactivation de la base de données spatiale (gseDisableDB) 3. Activation de la base de données spatiale (gseEnableDB)	1. Il s'agit de la première étape requise pour utiliser DB2 Extension Spatiale. Une base de données activée pour des opérations spatiales est dotée d'un ensemble de types spatiaux, d'un ensemble de fonctions spatiales, d'un ensemble de prédicats spatiaux, d'un nouveau type d'index et d'un ensemble de tables et de vues d'administration. 2. Cette étape intervient généralement lorsque vous avez activé les fonctionnalités spatiales pour une base de données incorrecte. Lorsque vous désactivez une base de données spatiale, vous supprimez un ensemble de types spatiaux, un ensemble de fonctions spatiales, un ensemble de prédicats spatiaux, un nouveau type d'index et un ensemble de tables et de vues d'administration. **Remarque :** La désactivation de la base de données échouera si des objets dépendant d'objets générés par la procédure d'activation de la base de données ont été créés. Par exemple, la création d'une table dotée d'une colonne spatiale de type ST_Point empêche la désactivation de la base de données. En effet, la table dépend du type ST_Point qui doit être supprimé dans le cadre de la procédure de désactivation de la base de données. 3. Identique au 1.

Tableau 5. Programme exemple DB2 Extension Spatiale (suite)

Etapes du programme exemple	Action	Description
Enregistrement des systèmes de références spatiales	1. Enregistrement du système de références spatiales associé à la colonne LOCATION de la table CUSTOMERS (gseEnableSref) 2. Enregistrement du système de références spatiales associé à la colonne LOCATION de la table OFFICES (gseEnableSref) 3. Désenregistrement du système de références spatiales associé à la colonne LOCATION de la table OFFICES (gseDisableSref) 4. Nouvel enregistrement du système de références spatiales associé aux colonnes ZONE de la table OFFICES (gseEnableSref)	1. Cette étape définit un nouveau système de références spatiales (SRS) destiné à l'interprétation des données spatiales de la table CUSTOMERS. Un système de références spatiales contient des données géométriques dans un format stockable dans une colonne de base de données activée pour les opérations spatiales. Une fois ce système enregistré pour une couche spécifique, les coordonnées applicables à cette couche peuvent être stockées dans la colonne associée de la table CUSTOMERS. 2. Cette étape définit un nouveau système de références spatiales (SRS) destiné à l'interprétation des données spatiales de la table OFFICES. Chaque couche de la table doit être associée à un système de références spatiales. Les couches de la table OFFICES devront éventuellement être associées à un système différent de celui de la couche de la table CUSTOMERS. 3. Cette étape intervient si vous spécifiez des paramètres de système de références spatiales incorrects pour la couche ou la colonne spatiale concernée. Lorsque vous désenregistrez un système de ce type pour la couche de la table OFFICES, vous supprimez la définition ainsi que les paramètres associés. 4. Cette étape définit un nouveau système de références spatiales (SRS) destiné à l'interprétation des données spatiales de la table OFFICES.

Tableau 5. Programme exemple DB2 Extension Spatiale (suite)

Etapes du programme exemple	Action	Description
Création des tables spatiales	1. Modification de la table CUSTOMERS par l'ajout de la colonne LOCATION (gseSetupTables) 2. Création de la table OFFICES (gseSetupTables)	1. La table CUSTOMERS contient les données commerciales stockées dans la base de données depuis plusieurs années. L'instruction ALTER TABLE ajoute une nouvelle colonne (LOCATION) de type ST_Point. Cette colonne sera peuplée en géocodant les colonnes d'adresse lors d'une étape ultérieure. 2. La table OFFICES comprend, parmi d'autres données, le secteur de vente de chaque succursale d'une compagnie d'assurances. La table entière sera peuplée avec des données d'attribut provenant d'une base de données non DB2 lors d'une étape ultérieure. Cette étape requiert l'importation de données d'attribut dans la table OFFICES à partir d'un fichier SHAPE.
Enregistrement des couches spatiales	1. Enregistrement de la colonne LOCATION de la table CUSTOMERS en tant que couche (gseRegisterLayer) 2. Enregistrement de la colonne ZONE de la table OFFICES en tant que couche (gseRegisterLayer)	Ces étapes permettent d'enregistrer les colonnes LOCATION et ZONE en tant que couches dans DB2 Extension Spatiale. Avant qu'une colonne spatiale puisse être peuplée ou que des utilitaires DB2 Extension Spatiale (géocodeur, etc.) puissent y accéder, vous devez l'enregistrer en tant que couche.

Tableau 5. Programme exemple DB2 Extension Spatiale (suite)

Etapes du programme exemple	Action	Description
Peuplement des couches spatiales	1. Géocodage des données d'adresse pour la colonne LOCATION de la table CUSTOMERS (gseRunGC) 2. Chargement de la table OFFICES en mode APPEND (gseImportShape) 3. Chargement de la table HAZARD_ZONE en mode CREATE (gseImportShape)	1. Cette étape permet de géocoder les données en traitement par lots en appelant le géocodeur. Le géocodage en traitement par lots intervient généralement lorsqu'une partie importante de la table a besoin d'être géocodée ou re-géocodée. 2. Cette étape charge dans la table OFFICES les données spatiales existant en tant que fichier SHAPE. Puisque la table OFFICES existe déjà et que la couche OFFICES/ZONE est enregistrée, l'utilitaire de chargement ajoute les nouveaux enregistrements à la fin d'une table existante. 3. Cette étape charge dans la couche HAZARD_ZONE les données spatiales existant en tant que fichier SHAPE. La table et la couche n'existant pas, l'utilitaire de chargement crée la table et enregistre la couche avant de charger les données.
Activation des index spatiaux	1. Activation de l'index spatial pour la colonne LOCATION de la table CUSTOMERS (gseEnableIdx) 2. Activation de l'index spatial pour la colonne ZONE de la table OFFICES (gseEnableIdx) 3. Activation de l'index spatial pour la colonne LOCATION de la table OFFICES (gseEnableIdx) 4. Activation de l'index spatial pour la colonne BOUNDARY de la table HAZARD_ZONE (gseEnableIdx)	Ces étapes activent l'index spatial pour les tables CUSTOMERS, OFFICES et HAZARD_ZONE.

Tableau 5. Programme exemple DB2 Extension Spatiale (suite)

Etapes du programme exemple	Action	Description
Activation du géocodage automatique	1. Activation du géocodage automatique pour les colonnes LOCATION et ADDRESS de la table CUSTOMERS (gseEnableAutoGC)	Cette étape active l'appel automatique du géocodeur. L'utilisation du géocodage automatique entraîne la synchronisation des colonnes LOCATION et ADDRESS de la table CUSTOMERS l'une par rapport à l'autre pour les opérations d'insertion et de mise à jour ultérieures.
Insertion/mise à jour de la table CUSTOMERS	1. Insertion d'enregistrements comportant une rue différente (gseInsDelUpd) 2. Mise à jour d'enregistrements avec une nouvelle adresse (gseInsDelUpd)	Ces étapes illustrent une insertion et une mise à jour effectuées sur la colonne LOCATION de la table CUSTOMERS. Une fois le géocodage automatique activé, une information de la colonne ADDRESS est automatiquement géocodée lorsqu'elle est insérée ou mise à jour dans la colonne LOCATION. Ce processus a été activé à l'étape précédente.
Désactivation du géocodage automatique	1. Désactivation du géocodage automatique pour la couche CUSTOMERS (gseDisableAutoGC) 2. Désactivation de l'index spatial pour la couche CUSTOMERS (gseDisableIdxCustomersLayer)	Ces étapes désactivent l'appel automatique du géocodeur et de l'index spatial en vue de la prochaine étape (celle-ci concerne le nouveau géocodage de la totalité de la table CUSTOMERS). Si vous chargez une grande quantité de données géographiques, il est recommandé de désactiver l'index spatial avant de charger les données, puis de l'activer de nouveau une fois ce chargement effectué.
Nouveau géocodage de la table CUSTOMERS	1. Nouveau géocodage de la couche CUSTOMERS avec un niveau de précision inférieur, 90 % au lieu 100 % (gseRunGC) 2. Réactivation de l'index spatial pour la couche CUSTOMERS (gseEnableIdx) 3. Réactivation du géocodage automatique avec un niveau de précision inférieur, 90 % au lieu de 100 % (gseEnableAutoGC)	Ces étapes permettent d'exécuter de nouveau le géocodeur en traitement par lots avec un degré de précision différent, et de réactiver l'index spatial et le géocodage automatique. Cette action est recommandée lorsque l'administrateur spatial remarque un taux d'échec élevé lors du processus de géocodage. Si la valeur de la précision est de 100 %, le géocodage d'une adresse n'aboutira pas car le système ne trouvera pas d'adresse correspondante dans les données de référence. En réduisant le niveau de précision, le géocodeur a plus de chances d'en trouver. Une fois la table à nouveau géocodée en traitement par lots, le géocodage automatique et l'index spatial sont tous deux réactivés afin de faciliter la mise à jour incrémentielle de la colonne et de l'index spatiaux lors des insertions et des mises à jour ultérieures.

Tableau 5. Programme exemple DB2 Extension Spatiale (suite)

Etapes du programme exemple	Action	Description
Création d'une vue et enregistrement de ses colonnes spatiales en tant que couches de vue	1. Création d'une vue, HIGHRISK_CUSTOMERS, obtenue à partir de la jointure des tables CUSTOMERS et HAZARD_ZONE (gseCreateView) 2. Enregistrement des colonnes spatiales de la vue en tant que couches de vue (gseRegisterLayer)	Ces étapes permettent de créer une vue et d'enregistrer les colonnes spatiales qu'elle contient en tant que couches de vue.
Analyse spatiale	1. Identification de la distance moyenne existant entre les clients et chaque grand magasin. 2. Détermination de la prime d'assurance et du revenu moyens pour chaque succursale (ST_Within) 3. Identification des clients non assurés par les succursales existantes (ST_Within) 4. Identification du nombre de zones à risque que chevauchent la zone couverte par chaque succursale (ST_Overlaps) 5. Détermination de la succursale la plus proche de l'habitation d'un client déterminé en supposant que la succursale est située au centre de la zone qu'elle couvre (ST_Distance, ST_Centroid) 6. Identification des clients dont l'habitation est proche du périmètre d'une zone à risque (ST_Buffer, ST_Overlaps) 7. Identification de tous les clients à haut risque couverts par une succursale déterminée (Toutes ces étapes recourent à gseRunSpatialQueries)	Ces étapes réalisent une analyse spatiale par l'utilisation de fonctions et prédicats spatiaux intégrés dans du langage SQL DB2. L'optimiseur de requêtes DB2 exploite l'index spatial associé aux colonnes spatiales pour améliorer les performances des requêtes dès que cela s'avère possible.

Tableau 5. Programme exemple DB2 Extension Spatiale (suite)

Etapes du programme exemple	Action	Description
Exportation de couches spatiales dans des fichiers	Exportation de la couche highRiskCustomers (gseExportShape)	L'étape illustre un exemple d'exportation du résultat de votre requête dans un fichier SHAPE. L'exportation de ce résultat dans un autre format de fichier permet d'utiliser les informations à l'aide d'un outil non IBM (par exemple, ESRI ArcInfo).

Partie 2. Informations de référence

Chapitre 9. Procédures mémorisées

Le présent chapitre décrit les procédures mémorisées qui vous permettent de créer un système d'informations géographiques (SIG) avec DB2 Extension Spatiale. Lorsque vous activez et utilisez DB2 Extension Spatiale à partir du Centre de contrôle, vous appelez implicitement ces procédures. Par exemple, quand vous cliquez sur **OK** dans une fenêtre DB2 Extension Spatiale, DB2 appelle automatiquement la ou les procédure(s) associée(s) à cette fenêtre. Par contre, vous pouvez appeler les procédures mémorisées de manière explicite dans un programme d'application. Il est alors recommandé d'inclure le fichier d'en-tête db2gse.h dans ce type de programme. En effet, il contient les macro-définitions des constantes que vous affectez aux paramètres des procédures mémorisées. Sous AIX, ce fichier réside dans le répertoire $DB2INSTANCE/sqllib/include/. Sous Windows NT, il est stocké dans le répertoire %DB2PATH%\include\.

Attention :
Toutes les constantes des chaînes de caractères, associées aux paramètres d'entrée des procédures mémorisées prennent en compte la distinction entre majuscules et minuscules. Pour connaître les paramètres nécessitent ces constantes, reportez-vous aux tableaux présentées dans ce chapitre.

Pour pouvoir appeler une procédure mémorisée, implicitement ou explicitement, vous devez tout d'abord vous connecter à la base de données dans laquelle est installé DB2 Extension Spatiale. La première procédure mémorisée que vous utilisez est db2gse.gse_enable_db. Elle active la base de données pour les opérations spatiales. Vous ne pouvez appeler les autres procédures de ce type qu'après l'activation de la base de données.

Les implémentations des procédures mémorisées sont archivées dans la bibliothèque db2gse du serveur DB2 Extension Spatiale.

Les listes suivantes répertorient les procédures mémorisées par nom ou en fonction des tâches qu'elles exécutent. La première liste les classe par nom :

- «db2gse.gse_enable_idx» à la page 81
- «db2gse.gse_enable_sref» à la page 84
- «db2gse.gse_export_shape» à la page 87
- «db2gse.gse_import_sde» à la page 89
- «db2gse.gse_import_shape» à la page 91
- «db2gse.gse_register_gc» à la page 93
- «db2gse.gse_register_layer» à la page 95
- «db2gse.gse_run_gc» à la page 102
- «db2gse.gse_unregist_gc» à la page 104
- «db2gse.gse_unregist_layer» à la page 105

La seconde liste classe les tâches exécutées par les procédures mémorisées.

- Création d'un index associé à une colonne spatiale (voir «db2gse.gse_enable_idx» à la page 81).
- Création d'un système de références spatiales (voir «db2gse.gse_enable_sref» à la page 84).
- Désactivation d'un géocodeur pour qu'il ne synchronise pas automatiquement les colonnes spatiales par rapport aux colonnes d'attribut correspondantes (voir «db2gse.gse_disable_autogc» à la page 72).
- Désactivation de la prise en charge des opérations spatiales dans une base de données (voir «db2gse.gse_disable_db» à la page 74).
- Suppression d'un système de références spatiales (voir «db2gse.gse_disable_sref» à la page 75).
- Activation d'une base de données pour la prise en charge des opérations spatiales (voir «db2gse.gse_enable_db» à la page 80).
- Activation d'un géocodeur pour qu'il synchronise automatiquement les colonnes spatiales par rapport aux colonnes d'attribut correspondantes (voir «db2gse.gse_enable_autogc» à la page 76).
- Exportation d'une couche et de la table associée dans un fichier SHAPE (voir «db2gse.gse_export_shape» à la page 87).
- Importation d'une couche et de la table associée à partir d'un fichier de transfert ESRI_SDE (voir «db2gse.gse_import_sde» à la page 89).
- Importation d'une couche et de la table associée à partir d'un fichier SHAPE (voir «db2gse.gse_import_shape» à la page 91).
- Enregistrement d'un géocodeur autre que le géocodeur par défaut (voir «db2gse.gse_register_gc» à la page 93).
- Enregistrement d'une colonne spatiale en tant que couche (voir «db2gse.gse_register_layer» à la page 95).
- Exécution d'un géocodeur en traitement par lots (voir «db2gse.gse_unregist_gc» à la page 104).

- Désenregistrement d'un géocodeur autre que le géocodeur par défaut (voir «db2gse.gse_unregist_layer» à la page 105).
- Désenregistrement d'une couche (voir «db2gse.gse_unregist_layer» à la page 105).

Pour plus d'informations sur les différentes possibilités d'ordonnancement de ces tâches, reportez-vous au «Chapitre 1. Présentation de DB2 Extension Spatiale» à la page 3, et au «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

db2gse.gse_disable_autogc

Cette procédure mémorisée permet de supprimer ou de désactiver temporairement les déclencheurs qui ont pour fonction de synchroniser une colonne spatiale par rapport à la ou aux colonne(s) d'attribut qui lui sont associées. Par exemple, il est recommandé de désactiver les déclencheurs pendant le géocodage en traitement par lots des valeurs contenues dans les colonnes d'attributs. Pour en savoir plus sur ce sujet, reportez-vous à la section «Présentation du géocodage» à la page 41.

Pour un exemple du code permettant d'appeler cette procédure mémorisée, reportez-vous à la fonction C gseDisableAutoGc dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer de l'autorisation appropriée octroyée sous la forme de droits, d'un privilège ou d'un ensemble de privilèges ; il s'agit en l'occurence de :

- droits SYSADM ou DBADM sur la base de données contenant la table sur laquelle sont définis les déclencheurs à supprimer ou à désactiver temporairement, ou
- privilège CONTROL sur cette table,
- privilèges ALTER, SELECT et UPDATE sur cette table.

Paramètres d'entrée

Tableau 6. Paramètres d'entrée de la procédure mémorisée db2gse.gse_disable_autogc.

Nom	Type de données	Description
operMode	SMALLINT	Indique si les déclencheurs doivent être supprimés ou temporairement désactivés. Ce paramètre ne peut pas prendre la valeur NULL. **Commentaire :** Pour supprimer des déclencheurs, utilisez la macro GSE_AUTOGC_DROP. Pour les désactiver temporairement, recourez à la macro GSE_AUTOGC_INVALIDATE. Pour déterminer les valeurs associées à ces macros, consultez le fichier db2gse.h. Sous AIX, ce fichier se trouve dans le répertoire $DB2INSTANCE/sqllib/include/. Sous Windows NT, il est stocké dans le répertoire %DB2PATH%\include\.

Tableau 6. Paramètres d'entrée de la procédure mémorisée db2gse.gse_disable_autogc. (suite)

Nom	Type de données	Description
layerSchema	VARCHAR(30)	Nom du schéma auquel appartient la table ou la vue spécifiée dans le paramètre layerTable. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si vous n'affectez pas de valeur au paramètre layerSchema, il prendra par défaut l'ID utilisateur sous lequel est appelée la procédure mémorisée db2gse.gse_disable_autogc.
layerTable	VARCHAR(128)	Nom de la table sur laquelle sont définis les déclencheurs que vous voulez supprimer ou désactiver temporairement. Ce paramètre ne peut pas prendre la valeur NULL.
layerColumn	VARCHAR(128)	Nom de la colonne activée pour les opérations spatiales et mise à jour par les déclencheurs que vous voulez supprimer ou désactiver temporairement. Ce paramètre ne peut pas prendre la valeur NULL.

Paramètres de sortie

Tableau 7. Paramètres de sortie de la procédure mémorisée db2gse.gse_disable_autogc.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_disable_db

Cette procédure mémorisée permet de supprimer des ressources utilisées par DB2 Extension Spatiale pour le stockage des données spatiales et la prise en charge des opérations effectuées sur ces données.

Le but de cette procédure est de vous aider à résoudre des incidents qui surviennent après l'activation de la base de données pour les opérations spatiales mais *avant* que vous n'y ajoutiez des colonne ou des données de table spatiales. Par exemple, supposons qu'après avoir activé une base de données pour les opérations spatiales, vous décidiez d'utiliser DB2 Extension Spatiale avec une autre base de données. Tant que vous n'avez pas défini de colonnes spatiales ou importé des données spatiales, vous pouvez appeler cette procédure mémorisée pour supprimer toutes les ressources spatiales de la première base de données.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseDisableDB dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer des droits SYSADM ou DBADM sur la base de données d'où des ressources DB2 Extension Spatiale doivent être supprimées.

Paramètres de sortie

Tableau 8. Paramètres de sortie de la procédure mémorisée db2gse.gse_disable_db.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_disable_sref

Cette procédure mémorisée permet de supprimer un système de références spatiales. Lors de son traitement, des informations sur le système de références sont supprimées de la vue du catalogue DB2GSE.SPATIAL_REF_SYS. Pour plus d'informations sur cette vue, reportez-vous à la section «DB2GSE.SPATIAL_REF_SYS» à la page 117.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseDisableSref dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

Néant

Paramètres d'entrée

Tableau 9. Paramètre d'entrée de la procédure mémorisée db2gse.gse_disable_sref.

Nom	Type de données	Description
srId	INTEGER	Identificateur numérique associé au système de références spatiales à supprimer. Ce paramètre ne peut pas prendre la valeur NULL.

Paramètres de sortie

Tableau 10. Paramètres de sortie de la procédure mémorisée db2gse.gse_disable_sref.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

Restriction

Avant de supprimer un système de références spatiales, vous devez désenregistrer toutes les couches qui l'utilisent. Si elles ne sont pas toutes désenregistrées, la requête de suppression du système de références spatiales est rejetée.

db2gse.gse_enable_autogc

Cette procédure mémorisée permet d'effectuer les tâches suivantes :

- Création de déclencheurs qui permettent de synchroniser une colonne spatiale par rapport à la ou aux colonne(s) d'attribut qui lui sont associées. Lors de chaque insertion ou de mise à jour de valeurs dans la ou les colonne(s) d'attribut, un déclencheur appelle un géocodeur enregistré afin qu'il convertisse ces valeurs et place les données ainsi obtenues dans la colonne spatiale.
- Réactivation des déclencheurs après leur désactivation temporaire.
- Détermination de la fonction à utiliser pour géocoder les valeurs insérées ou mises à jour.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseEnableAutoGC dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer de l'autorisation appropriée octroyée sous la forme de droits, d'un privilège ou d'un ensemble de privilèges ; il s'agit en l'occurence de :

- droits SYSADM ou DBADM sur la base de données contenant la table sur laquelle sont définis les déclencheurs créés par cette procédure mémorisée ou
- privilège CONTROL sur cette table,
- privilèges ALTER, SELECT et UPDATE sur cette table.

Paramètres d'entrée

Tableau 11. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_autogc.

Nom	Type de données	Description
operMode	SMALLINT	Indique si les déclencheurs qui lancent le géocodage doivent être créés ou réactivés après avoir été temporairement désactivés. Ce paramètre ne peut pas prendre la valeur NULL. **Commentaire :** Pour créer des déclencheurs, utilisez la macro GSE_AUTOGC_CREATE. Pour les réactiver, recourez à la macro GSE_AUTOGC_RECREATE. Pour déterminer les valeurs associées à ces macros, consultez le fichier db2gse.h. Sous AIX, ce fichier se trouve dans le répertoire $DB2INSTANCE/sqllib/include/. Sous Windows NT, il est stocké dans le répertoire %DB2PATH%\include\.
layerSchema	VARCHAR(30)	Nom du schéma auquel appartient la table spécifiée dans le paramètre layerTable. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si vous n'affectez pas de valeur au paramètre layerSchema, il prendra par défaut l'ID utilisateur sous lequel est appelée procédure mémorisée db2gse.gse_enable_autogc.
layerTable	VARCHAR(128)	Nom de la table concernée par l'exécution des déclencheurs créés ou réactivés par cette procédure mémorisée. Ce paramètre ne peut pas prendre la valeur NULL.
layerColumn	VARCHAR(128)	Nom de la colonne spatiale mise à jour par les déclencheurs créés ou réactivés par cette procédure mémorisée. Ce paramètre ne peut pas prendre la valeur NULL.

Tableau 11. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_autogc. (suite)

Nom	Type de données	Description
gcId	INTEGER	Identificateur du géocodeur qui doit être appelé par les déclencheurs d'insertion et de mise à jour créés ou réactivés par cette procédure mémorisée. Ce paramètre ne peut pas prendre la valeur NULL si le paramètre operMode est défini par GSE_AUTOGC_CREATE. Il peut la prendre si operMode a la valeur GSE_AUTOGC_RECREATE.
precisionLevel	INTEGER	Degré de correspondance requis entre les données source et les données de référence pour que le géocodeur traite les données source avec succès. Ce paramètre ne peut pas prendre la valeur NULL si le paramètre operMode est défini par GSE_AUTOGC_CREATE. Il peut la prendre si operMode a la valeur GSE_AUTOGC_RECREATE. **Commentaire :** Le niveau de précision peut aller de 1 à 100 %.
vendorSpecific	VARCHAR(256)	Informations techniques données par le fournisseur ; par exemple, le chemin d'accès et le nom du fichier qu'il utilise pour la définition des paramètres. Ce paramètre ne peut pas prendre la valeur NULL si le paramètre operMode est défini par GSE_AUTOGC_CREATE. Il peut la prendre si operMode a la valeur GSE_AUTOGC_RECREATE.

Paramètres de sortie

Tableau 12. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_autogc.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

Restrictions

- Le paramètre layerColumn doit faire référence à une colonne enregistrée en tant que couche de table.
- Si le paramètre operMode a la valeur GSE_AUTOGC_CREATE, vous devez affecter l'identificateur d'un géocodeur enregistré au paramètre gcId.

db2gse.gse_enable_db

Cette procédure mémorisée permet d'affecter à une base de données les ressources nécessaires au stockage des données spatiales et à la prise en charge des opérations effectuées sur ces données. Ces ressources comprennent des types de données spatiales, un type d'index spatial, des tables et vues de catalogue, des fonctions fournies et d'autres procédures mémorisées.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseEnableDB dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer des droits SYSADM ou DBADM sur la base de données qui est activée.

Paramètres de sortie

Tableau 13. Paramètres de sortie de la procédure mémorisée db2gse.gse_enable_db.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_enable_idx

Cette procédure mémorisée permet de créer un index associé à une colonne spatiale.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseEnableIdx dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer des droits ou privilèges suivants :

- droits SYSADM ou DBADM sur la base de données contenant la table à laquelle doit s'appliquer l'index activé ou
- privilège CONTROL ou INDEX sur cette table.

Paramètres d'entrée

Tableau 14. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_idx.

Nom	Type de données	Description
layerSchema	VARCHAR(30)	Nom du schéma auquel appartient la table spécifiée dans le paramètre layerTable. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si vous n'affectez pas de valeur au paramètre layerSchema, il prendra par défaut l'ID utilisateur sous lequel est appelée la procédure mémorisée db2gse.gse_enable_idx.
layerTable	VARCHAR(128)	Nom de la table sur laquelle doit être défini l'index que vous voulez créer. Ce paramètre ne peut pas prendre la valeur NULL.
layerColumn	VARCHAR(128)	Nom de la colonne activée pour les opérations spatiales, sur laquelle porte la recherche réalisée avec l'index que vous créez. Ce paramètre ne peut pas prendre la valeur NULL.

Tableau 14. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_idx. (suite)

Nom	Type de données	Description
indexName	VARCHAR(128)	Nom de l'index à créer. Ce paramètre ne peut pas prendre la valeur NULL. **Commentaire :** Ne spécifiez pas un nom de schéma. DB2 Extension Spatiale affecte automatiquement l'index au schéma précisé par le paramètre layerSchema.
gridSize1	DOUBLE	Nombre indiquant la trame de la grille d'index la plus serrée possible. Ce paramètre ne peut pas prendre la valeur NULL.
gridSize2	DOUBLE	Nombre qui indique (1) qu'il n'y aura pas de seconde grille associée à cet index ou (2) quelle doit être la granularité de la seconde grille. Ce paramètre peut prendre la valeur NULL. **Commentaire :** S'il ne doit pas y avoir de seconde grille, spécifiez 0. Dans le cas contraire, la trame de cette grille doit être plus large que celle de la grille définie par le paramètre gridSize1.
gridSize3	DOUBLE	Nombre qui indique (1) qu'il n'y aura pas de troisième grille associée à cet index ou (2) quelle doit être la granularité de la troisième grille. Ce paramètre peut prendre la valeur NULL. **Commentaire :** S'il ne doit pas y avoir de troisième grille, spécifiez 0. Dans le cas contraire, la trame de cette grille doit être plus large que celle de la grille définie par le paramètre gridSize2.

Paramètres de sortie

Tableau 15. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_idx.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.

Tableau 15. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_idx. (suite)

Nom	Type de données	Description
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_enable_sref

Cette procédure mémorisée permet de définir le mode de conversion des nombres négatifs et décimaux d'un système de coordonnées déterminé en nombres entiers positifs, et ce, en vue de leur stockage par DB2 Extension Spatiale. L'ensemble des valeurs spécifiées par vos soins s'appelle un *système de références spatiales*. Lors du traitement de cette procédure, des informations sur le système de référence sont ajoutées à la vue du catalogue DB2GSE.SPATIAL_REF_SYS. Pour plus d'informations sur cette vue, reportez-vous à la section «DB2GSE.SPATIAL_REF_SYS» à la page 117.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseEnableSref dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

Néant

Paramètres d'entrée

Tableau 16. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_sref.

Nom	Type de données	Description
srId	INTEGER	Identificateur numérique associé au système de références spatiales. Ce paramètre ne peut pas prendre la valeur NULL. **Commentaire :** Cet identificateur doit être unique au sein de la base de données activée pour les opérations spatiales.
srName	VARCHAR(64)	Description succincte du système de références spatiales. Ce paramètre ne peut pas prendre la valeur NULL. **Commentaire :** Cette description doit être unique au sein de la base de données activée pour les opérations spatiales.
falsex	DOUBLE	Nombre qui, soustrait d'une valeur d'abscisse (coordonnée X) négative, donne un nombre non négatif, autrement dit positif ou égal à zéro. Ce paramètre ne peut pas prendre la valeur NULL.

Tableau 16. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_sref. (suite)

Nom	Type de données	Description
falsey	DOUBLE	Nombre qui, soustrait d'une valeur d'ordonnée (coordonnée Y) négative, donne un nombre non négatif, autrement dit positif ou égal à zéro. Ce paramètre ne peut pas prendre la valeur NULL.
xyunits	DOUBLE	Nombre qui, multiplié par une abscisse (X) ou une ordonnée (Y) décimale, donne un entier stockable en tant qu'élément de donnée 32 bits. Ce paramètre ne peut pas prendre la valeur NULL.
falsez	DOUBLE	Nombre qui, soustrait d'une valeur de coordonnée Z négative, donne un nombre non négatif, autrement dit positif ou égal à zéro. Ce paramètre ne peut pas prendre la valeur NULL.
zunits	DOUBLE	Nombre qui, multiplié par une coordonnée Z décimale, donne un entier stockable en tant qu'élément de donnée 32 bits. Ce paramètre ne peut pas prendre la valeur NULL.
falsem	DOUBLE	Nombre qui, soustrait d'une mesure négative, donne un nombre non négatif, autrement dit positif ou égal à zéro. Ce paramètre ne peut pas prendre la valeur NULL.
munits	DOUBLE	Nombre qui, multiplié par une mesure décimale, donne un entier stockable en tant qu'élément de donnée 32 bits. Ce paramètre ne peut pas prendre la valeur NULL.

Tableau 16. Paramètres d'entrée de la procédure mémorisée db2gse.gse_enable_sref. (suite)

Nom	Type de données	Description
scId	INTEGER	Identificateur numérique du système de coordonnées dont est dérivé le système de références spatiales. Pour trouver cet identificateur, consultez la vue du catalogue DB2GSE.COORD_REF_SYS à la section «DB2GSE.COORD_REF_SYS» à la page 115. Ce paramètre ne peut pas prendre la valeur NULL.

Paramètres de sortie

Tableau 17. Paramètres de sortie de la procédure mémorisée db2gse.gse_enable_sref.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_export_shape

Cette procédure mémorisée permet d'exporter une couche et sa table associée vers un fichier SHAPE ou de créer un nouveau fichier SHAPE et d'y exporter une couche et sa table associée.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseExportShape dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer du privilège SELECT sur la table à exporter

Paramètres d'entrée

Tableau 18. Paramètres d'entrée de la procédure mémorisée db2gse.gse_export_shape.

Nom	Type de données	Description
layerSchema	VARCHAR(30)	Nom du schéma auquel appartient la table spécifiée dans le paramètre layerTable. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si vous n'affectez pas de valeur au paramètre layerSchema, il prendra par défaut l'ID utilisateur sous lequel est appelée la procédure mémorisée db2gse.gse_export_shape.
layerTable	VARCHAR(128)	Nom de la table à exporter. Ce paramètre ne peut pas prendre la valeur NULL.
layerColumn	VARCHAR(30)	Nom de la colonne enregistrée en tant que couche, que vous exportez. Ce paramètre ne peut pas prendre la valeur NULL.
fileName	VARCHAR(128)	Nom du fichier SHAPE vers lequel doit être exportée la couche spécifiée. Ce paramètre ne peut pas prendre la valeur NULL.

Tableau 18. Paramètres d'entrée de la procédure mémorisée db2gse.gse_export_shape. (suite)

Nom	Type de données	Description
whereClause	VARCHAR(1024)	Corps de la clause SQL WHERE. Il définit une restriction à appliquer au lot d'enregistrements qui doit être géocodé. La clause peut faire référence à toute colonne d'attribut appartenant à la table que vous exportez. Ce paramètre peut prendre la valeur NULL.

Paramètres de sortie

Tableau 19. Paramètres de sortie de la procédure mémorisée db2gse.gse_export_shape.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

Restriction

Vous ne pouvez exporter qu'une couche à la fois.

db2gse.gse_import_sde

Cette procédure mémorisée permet d'importer un fichier de transfert SDE dans une base de données activée pour les opérations spatiales. Elle peut fonctionner selon deux modes :

- Si le fichier de transfert SDE est destiné à une table existante qui comporte une colonne de couche enregistrée, DB2 Extension Spatiale charge les données du fichier dans la table.
- Si tel n'est pas le cas, DB2 Extension Spatiale crée une table dotée d'une colonne spatiale, enregistre cette dernière en tant que couche et charge les données du fichier dans la couche et les autres colonnes de la table.

Le système de références spatiales spécifié dans le fichier de transfert SDE est comparé à ceux enregistrés auprès de DB2 Extension Spatiale. S'il correspond à un de ces systèmes, les valeurs négatives et décimales contenues dans les données de transfert seront transformées, lors du chargement, selon la méthode indiquée par le système enregistré. S'il ne correspond à aucun des systèmes enregistrés, DB2 Extension Spatiale crée un nouveau système de références spatiales qui décrit les modifications à apporter.

Classe d'autorisation

En cas d'importation de données dans une table existante, l'ID utilisateur sous lequel la procédure mémorisée est appelée doit disposer des droits ou privilèges suivants :

- droits SYSADM ou DBADM sur la base de données contenant la table dans laquelle les données doivent être importées, ou
- privilège CONTROL sur cette table.

En cas d'importation de données dans une table qui doit être créée, l'ID utilisateur sous lequel la procédure mémorisée est appelée doit disposer des droits ou privilèges suivants :

- droits SYSADM ou DBADM sur la base de données contenant la table qui doit être créée.

Paramètres d'entrée

Tableau 20. Paramètres d'entrée de la procédure mémorisée db2gse.gse_import_sde.

Nom	Type de données	Description
layerSchema	VARCHAR(30)	Nom du schéma auquel appartient la table ou la vue spécifiée dans le paramètre layerTable. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si vous n'affectez pas de valeur au paramètre layerSchema, il prendra par défaut l'ID utilisateur sous lequel est appelée la procédure mémorisée db2gse.gse_import_sde.

Tableau 20. Paramètres d'entrée de la procédure mémorisée db2gse.gse_import_sde. (suite)

Nom	Type de données	Description
layerTable	VARCHAR(128)	Nom de la table dans laquelle les données de transfert SDE doivent être chargées. Ce paramètre ne peut pas prendre la valeur NULL.
layerColumn	VARCHAR(30)	Nom de la colonne enregistrée en tant que couche, dans laquelle doivent être chargées les données spatiales du fichier de transfert SDE. Ce paramètre ne peut pas prendre la valeur NULL.
fileName	VARCHAR(128)	Nom du fichier de transfert SDE à importer. Ce paramètre ne peut pas prendre la valeur NULL.
commitScope	INTEGER	Nombre d'enregistrements par point de contrôle. Ce paramètre peut prendre la valeur NULL.

Paramètres de sortie

Tableau 21. Paramètres de sortie de la procédure mémorisée db2gse.gse_import_sde.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_import_shape

Cette procédure mémorisée permet d'importer un fichier SHAPE dans une base de données activée pour les opérations spatiales. Elle peut fonctionner selon deux modes :

- Si le fichier SHAPE est destiné à une table existante qui comporte une colonne de couche enregistrée, DB2 Extension Spatiale charge les données du fichier dans la table.
- Si tel n'est pas le cas, DB2 Extension Spatiale crée une table dotée d'une colonne spatiale, enregistre cette dernière en tant que couche et charge les données du fichier dans la couche et les autres colonnes de la table.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseImportShape dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer des droits ou privilèges suivants :

- droits SYSADM ou DBADM sur la base de données contenant la table dans laquelle les données doivent être importées, ou
- privilège CONTROL sur cette table.

Paramètres d'entrée

Tableau 22. Paramètres d'entrée de la procédure mémorisée db2gse.gse_import_shape.

Nom	Type de données	Description
layerSchema	VARCHAR(30)	Nom du schéma auquel appartient la table ou la vue spécifiée dans le paramètre layerTable. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si vous n'affectez pas de valeur au paramètre layerSchema, il prendra par défaut l'ID utilisateur sous lequel est appelée la procédure mémorisée db2gse.gse_import_shape.
layerTable	VARCHAR(128)	Nom de la table dans laquelle le fichier SHAPE doit être chargé. Ce paramètre ne peut pas prendre la valeur NULL.

Tableau 22. Paramètres d'entrée de la procédure mémorisée db2gse.gse_import_shape. (suite)

Nom	Type de données	Description
layerColumn	VARCHAR(30)	Nom de la colonne enregistrée en tant que couche, dans laquelle les données du fichier SHAPE doivent être chargées. Ce paramètre ne peut pas prendre la valeur NULL.
fileName	VARCHAR(128)	Nom du fichier SHAPE à importer. Ce paramètre ne peut pas prendre la valeur NULL.
exceptionFile	VARCHAR(128)	Chemin d'accès et nom du fichier destiné au stockage des formes (SHAPE) dont l'importation a échoué. Il s'agit d'un nouveau fichier qui est généré au cours de l'exécution de la procédure db2gse.gse_import_shape. Ce paramètre ne peut pas prendre la valeur NULL.
srId	INTEGER	Identificateur du système de références spatiales à associer à la couche dans laquelle les données du fichier SHAPE doivent être chargées. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si l'identificateur n'est pas spécifié, les transformations internes s'effectueront selon la résolution maximale possible pour le fichier SHAPE.
commitScope	INTEGER	Nombre d'enregistrements par point de contrôle. Ce paramètre peut prendre la valeur NULL.

Paramètres de sortie

Tableau 23. Paramètres de sortie de la procédure mémorisée db2gse.gse_import_shape.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_register_gc

Cette procédure mémorisée permet d'enregistrer un géocodeur autre que le géocodeur par défaut. Pour savoir si un géocodeur a déjà été enregistré, consultez la vue du catalogue DB2GSE.SPATIAL_GEOCODER (décrite à la section «DB2GSE.SPATIAL_GEOCODER» à la page 117).

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer des droits SYSADM ou DBADM sur la base de données qui contient le géocodeur enregistré par cette procédure mémorisée.

Paramètres d'entrée

Tableau 24. Paramètres d'entrée de la procédure mémorisée db2gse.gse_register_gc.

Nom	Type de données	Description
gcId	INTEGER	Identificateur numérique du géocodeur que vous enregistrez. Ce paramètre ne peut pas prendre la valeur NULL. **Commentaire :** Cet identificateur doit être unique au sein de la base de données.
gcName	VARCHAR(64)	Description succincte du géocodeur que vous enregistrez. Ce paramètre ne peut pas prendre la valeur NULL. **Commentaire :** Cette description doit être une chaîne de caractères unique au sein de la base de données.
vendorName	VARCHAR(64)	Nom du fournisseur du géocodeur que vous enregistrez. Ce paramètre ne peut pas prendre la valeur NULL.
primaryUDF	VARCHAR(256)	Nom qualifié complet du géocodeur que vous enregistrez. Ce paramètre ne peut pas prendre la valeur NULL.

Tableau 24. Paramètres d'entrée de la procédure mémorisée db2gse.gse_register_gc. (suite)

Nom	Type de données	Description
precisionLevel	INTEGER	Degré de correspondance requis entre les données source et les données de référence pour que le géocodeur traite les données source avec succès. Ce paramètre ne peut pas prendre la valeur NULL. **Commentaire :** Le niveau de précision peut aller de 1 à 100 %.
vendorSpecific	VARCHAR(256)	Informations techniques données par le fournisseur ; par exemple, le chemin d'accès et le nom du fichier qu'il utilise pour la définition des paramètres. Ce paramètre peut prendre la valeur NULL.
geoArea	VARCHAR(256)	Zone géographique à géocoder. Ce paramètre peut prendre la valeur NULL.
description	VARCHAR(256)	Remarques provenant du fournisseur. Ce paramètre peut prendre la valeur NULL.

Paramètres de sortie

Tableau 25. Paramètres de sortie de la procédure mémorisée db2gse.gse_register_gc.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_register_layer

Cette procédure mémorisée permet d'enregistrer une colonne spatiale en tant que couche. Lors du traitement de cette procédure, des informations sur la couche sont ajoutées à la vue du catalogue DB2GSE.GEOMETRY_COLUMNS. Pour plus d'informations sur cette vue, reportez-vous à la section «DB2GSE.GEOMETRY_COLUMNS» à la page 116.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseRegisterLayer dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer des droits ou privilèges suivants :

- Pour une couche de table :
  - droits SYSADM ou DBADM sur la base de données contenant la table à laquelle appartient la couche, ou
  - privilège CONTROL ou ALTER sur cette table.
- Pour une couche de vue :
  - privilège SELECT sur la ou les table(s) de base qui contiennent (1) les données d'adresse à géocoder pour cette couche et (2) les données spatiales issues du géocodage.

Paramètres d'entrée

Tableau 26. Paramètres d'entrée de la procédure mémorisée db2gse.gse_register_layer.

Nom	Type de données	Description
layerSchema	INTEGER(30)	Nom du schéma auquel appartient la table ou la vue spécifiée dans le paramètre layerTable. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si vous n'affectez pas de valeur au paramètre layerSchema, il prendra par défaut l'ID utilisateur sous lequel est appelée la procédure mémorisée db2gse.gse_register_layer.
layerTable	VARCHAR(128)	Nom de la table ou de la vue contenant la colonne qui est enregistrée en tant que couche. Ce paramètre ne peut pas prendre la valeur NULL.

Tableau 26. Paramètres d'entrée de la procédure mémorisée db2gse.gse_register_layer. (suite)

Nom	Type de données	Description
layerColumn	VARCHAR(128)	Nom de la colonne qui est enregistrée en tant que couche. Si cette colonne n'existe pas, DB2 Extension Spatiale la crée. Ce paramètre ne peut pas prendre la valeur NULL.
layerTypeName	VARCHAR(64)	Type de données de la colonne qui est enregistrée en tant que couche. Vous devez indiquer le type de données en lettres majuscules, par exemple : `ST_POINT` Ce paramètre ne peut pas prendre la valeur NULL s'il s'agit d'une colonne de table qui doit être créée pendant l'exécution de la procédure mémorisée. Si tel n'est pas le cas, s'il s'agit d'une colonne existant déjà au sein d'une table ou d'une vue, ce paramètre peut prendre la valeur NULL.
srId	INTEGER	Identificateur du système de références spatiales associé à la couche concernée. Ce paramètre ne peut pas prendre la valeur NULL pour une couche de table. DB2 Extension Spatiale l'ignore lors de l'enregistrement d'une couche de vue.
geoSchema	VARCHAR(30)	Applicable en cas d'enregistrement d'une colonne de vue en tant que couche. Le paramètre geoSchema indique le schéma de la table sous-jacente de la vue à laquelle appartient la colonne. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de vue en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de table. **Commentaire :** Si vous n'affectez pas de valeur au paramètre geoSchema, il prendra par défaut la valeur du paramètre layerSchema.

Tableau 26. Paramètres d'entrée de la procédure mémorisée db2gse.gse_register_layer. (suite)

Nom	Type de données	Description
geoTable	VARCHAR(128)	Applicable en cas d'enregistrement d'une colonne de vue en tant que couche. Le paramètre geoTable indique le nom de la table sous-jacente de la vue à laquelle appartient la colonne. Ce paramètre ne peut pas prendre la valeur NULL en cas d'enregistrement d'une colonne de vue en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de table.
geoColumn	VARCHAR(128)	Applicable en cas d'enregistrement d'une colonne de vue en tant que couche. Le paramètre geoColumn indique le nom de la colonne de table sous-jacente de la colonne de vue en question. Ce paramètre ne peut pas prendre la valeur NULL en cas d'enregistrement d'une colonne de vue en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de table.
nAttributes	SMALLINT	Nombre de colonnes contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue.
attr1Name	VARCHAR(128)	Nom de la première colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Si vous envisagez d'utiliser le géocodeur par défaut, vous devez stocker les adresses par rue dans la colonne attr1Name.

Tableau 26. Paramètres d'entrée de la procédure mémorisée db2gse.gse_register_layer. (suite)

Nom	Type de données	Description
attr2Name	VARCHAR(128)	Nom de la seconde colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Si vous envisagez d'utiliser le géocodeur par défaut, vous devez stocker les noms de villes dans la colonne attr2Name.
attr3Name	VARCHAR(128)	Nom de la troisième colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Si vous envisagez d'utiliser le géocodeur par défaut, vous devez stocker les noms ou abréviations des états dans la colonne attr3Name.
attr4Name	VARCHAR(128)	Nom de la quatrième colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Si vous envisagez d'utiliser le géocodeur par défaut, vous devez stocker les codes postaux dans la colonne attr4Name.

Tableau 26. Paramètres d'entrée de la procédure mémorisée db2gse.gse_register_layer. (suite)

Nom	Type de données	Description
attr5Name	VARCHAR(128)	Nom de la cinquième colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Le géocodeur par défaut ignore la colonne Attr5Name.
attr6Name	VARCHAR(128)	Nom de la sixième colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Le géocodeur par défaut ignore la colonne Attr6Name.
attr7Name	VARCHAR(128)	Nom de la septième colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Le géocodeur par défaut ignore la colonne Attr7Name.

Tableau 26. Paramètres d'entrée de la procédure mémorisée db2gse.gse_register_layer. (suite)

Nom	Type de données	Description
attr8Name	VARCHAR(128)	Nom de la huitième colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Le géocodeur par défaut ignore la colonne Attr8Name.
attr9Name	VARCHAR(128)	Nom de la neuvième colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Le géocodeur par défaut ignore la colonne Attr9Name.
attr10Name	VARCHAR(128)	Nom de la dixième colonne contenant les données source qui doivent être géocodées pour cette couche. Ce paramètre peut prendre la valeur NULL en cas d'enregistrement d'une colonne de table en tant que couche. DB2 Extension Spatiale l'ignore lorsqu'il s'agit de l'enregistrement d'une couche de vue. Le géocodeur par défaut ignore la colonne Attr10Name.

Paramètres de sortie

Tableau 27. Paramètres de sortie de la procédure mémorisée db2gse.gse_register_layer.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

Restrictions

- Si vous enregistrez une colonne de vue en tant que couche, elle doit dépendre d'une colonne de table déjà enregistrée en tant que couche.
- Le nombre de colonnes contenant les données à géocoder pour la couche en cours d'enregistrement ne doit pas dépasser dix.

db2gse.gse_run_gc

Cette procédure mémorisée permet d'exécuter un géocodeur en traitement par lots. Pour plus d'informations sur cette tâche, reportez-vous à la section «Exécution du géocodeur en traitement par lots» à la page 43.

Pour un exemple du code requis pour appeler cette procédure mémorisée, reportez-vous à la fonction C gseRunGC dans le programme exemple. Pour plus d'informations sur ce programme, consultez le «Chapitre 8. Rédaction d'applications pour DB2 Extension Spatiale» à la page 59.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer des droits ou privilèges suivants :

- droits SYSADM ou DBADM sur la base de données contenant la table sur laquelle doit s'exécuter le géocodeur spécifié.
- privilège CONTROL ou UPDATE sur cette table.

Paramètres d'entrée

Tableau 28. Paramètres d'entrée de la procédure mémorisée db2gse.gse_run_gc.

Nom	Type de données	Description
layerSchema	VARCHAR(30)	Nom du schéma auquel appartient la table ou la vue spécifiée dans le paramètre layerTable. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si vous n'affectez pas de valeur au paramètre layerSchema, il prendra par défaut l'ID utilisateur sous lequel est appelée la procédure mémorisée db2gse.gse_run_gc.
layerTable	VARCHAR(128)	Nom de la table contenant la colonne dans laquelle les données géocodées doivent être insérées. Ce paramètre ne peut pas prendre la valeur NULL.
layerColumn	VARCHAR(128)	Nom de la colonne dans laquelle les données géocodées doivent être insérées. Ce paramètre ne peut pas prendre la valeur NULL.

Tableau 28. Paramètres d'entrée de la procédure mémorisée db2gse.gse_run_gc. (suite)

Nom	Type de données	Description
gcId	INTEGER	Identificateur du géocodeur à exécuter. Ce paramètre peut prendre la valeur NULL. Pour connaître les identificateurs des géocodeurs enregistrés, consultez la vue du catalogue DB2GSE.SPATIAL_GEOCODER.
precisionLevel	INTEGER	Degré de correspondance requis entre les données source et les données de référence pour que le géocodeur traite les données source avec succès. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Le niveau de précision peut aller de 1 à 100 %.
vendorSpecific	VARCHAR(256)	Informations techniques données par le fournisseur ; par exemple, le chemin d'accès et le nom du fichier qu'il utilise pour la définition des paramètres. Ce paramètre peut prendre la valeur NULL.
whereClause	VARCHAR(256)	Corps de la clause SQL WHERE. Il définit une restriction à appliquer au lot d'enregistrements qui doit être géocodé. La clause peut faire référence à toute colonne d'attribut appartenant à la table sur laquelle doit s'exécuter le géocodeur. Ce paramètre peut prendre la valeur NULL.
commitScope	INTEGER	Nombre d'enregistrements par point de contrôle. Ce paramètre peut prendre la valeur NULL.

Paramètres de sortie

Tableau 29. Paramètres de sortie de la procédure mémorisée db2gse.gse_run_gc.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_unregist_gc

Cette procédure mémorisée permet de désenregistrer un géocodeur autre que le géocodeur par défaut.

Pour trouver des informations sur le géocodeur à désenregistrer, consultez la vue du catalogue DB2GSE.SPATIAL_GEOCODER à la section «DB2GSE.SPATIAL_GEOCODER» à la page 117.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer des droits SYSADM ou DBADM sur la base de données qui contient le géocodeur à désenregistrer.

Paramètres d'entrée

Tableau 30. Paramètre d'entrée de la procédure mémorisée db2gse.gse_unregist_gc.

Nom	Type de données	Description
gcId	INTEGER	Identificateur du géocodeur à désenregistrer. Ce paramètre ne peut pas prendre la valeur NULL.

Paramètres de sortie

Tableau 31. Paramètres de sortie de la procédure mémorisée db2gse.gse_unregist_gc.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

db2gse.gse_unregist_layer

Cette procédure mémorisée permet de désenregistrer une couche. Pour ce faire, elle effectue les opérations suivantes :

- Retrait de la définition de la couche dans les tables du catalogue DB2 Extension Spatiale.
- Suppression de la contrainte de vérification placée par DB2 Extension Spatiale sur la table de base de la couche afin que les données spatiales de la dite couche respectent les conditions requises par le système de références spatiales auquel elle est associée.
- Suppression des déclencheurs utilisés pour mettre à jour la colonne spatiale dès l'ajout, la modification ou la suppression d'une donnée d'adresse.

Lors du traitement de cette procédure, les informations sur la couche sont supprimées de la métavue DB2GSE.GEOMETRY_COLUMNS. Pour plus d'informations sur cette vue, reportez-vous à la section «DB2GSE.GEOMETRY_COLUMNS» à la page 116.

Classe d'autorisation

L'ID utilisateur sous lequel est appelée la procédure mémorisée doit disposer des droits ou privilèges suivants :

- Pour une couche de table :
  - droits SYSADM ou DBADM sur la base de données contenant la table de base de la couche, ou
  - privilège CONTROL ou ALTER sur cette table.
- Pour une couche de vue :
  - privilège SELECT sur la ou les table(s) de base qui contiennent (1) les données d'adresse géocodées pour cette couche et (2) les données spatiales issues du géocodage.

Paramètres d'entrée

Tableau 32. Paramètres d'entrée de la procédure mémorisée db2gse.gse_unregist_layer.

Nom	Type de données	Description
layerSchema	VARCHAR(30)	Nom du schéma auquel appartient la table spécifiée dans le paramètre layerTable. Ce paramètre peut prendre la valeur NULL. **Commentaire :** Si vous n'affectez pas de valeur au paramètre layerSchema, il prendra par défaut l'ID utilisateur sous lequel est appelée la procédure mémorisée db2gse.gse_unregister_layer.

Tableau 32. Paramètres d'entrée de la procédure mémorisée db2gse.gse_unregist_layer. (suite)

Nom	Type de données	Description
layerTable	VARCHAR(128)	Nom de la table contenant la colonne spécifiée par le paramètre layerColum. Ce paramètre ne peut pas prendre la valeur NULL.
layerColumn	VARCHAR(128)	Nom de la colonne spatiale définie en tant que couche et que vous voulez désenregistrer. Ce paramètre ne peut pas prendre la valeur NULL.

Paramètres de sortie

Tableau 33. Paramètres de sortie de la procédure mémorisée db2gse.gse_unregist_layer.

Nom	Type de données	Description
msgCode	INTEGER	Code associé aux messages susceptibles d'être renvoyés par le demandeur de la procédure mémorisée.
Reservé	VARCHAR(1024)	Message d'erreur complet, tel que généré sur le serveur DB2 Extension Spatiale.

Restriction

Si une colonne de vue définie en tant que couche de vue dépend d'une colonne de table définie en tant que couche de table, vous ne pouvez pas désenregistrer la couche de table tant que vous n'avez pas effectué cette opération sur la couche de vue.

Chapitre 10. Messages

Le présent chapitre décrit les messages que DB2 Extension Spatiale renvoie aux utilisateurs. Chaque message est doté d'un identificateur. Les identificateurs qui se terminent par la lettre E désignent les messages d'erreur, ceux finissant par la lettre W, les avertissements, et ceux dont la dernière lettre est un I sont destinés aux informations d'ordre général.

DBA7200E 10 colonnes au maximum peuvent être sélectionnées comme entrée d'un géocodeur.

Explications : Le nombre de colonnes qui peuvent être sélectionnées en tant qu'entrée pour un géocodeur est limité à 10.

Réponse de l'utilisateur : Transférez des noms de colonnes de la liste Colonnes sélectionnées dans la liste Colonnes disponibles jusqu'à ce que la première liste ne contienne plus que 10 noms ou moins.

DBA7201E La base de données n'est pas activée pour effectuer des opérations d'extension spatiale.

Explications : Pour que vous puissiez utiliser l'extension spatiale, la base de données doit être activée pour les opérations spatiales.

Réponse de l'utilisateur : Cliquez avec le bouton droit de la souris sur la base de données et sélectionnez l'option Extension Spatiale —> Activation à partir des menus.

GSE0000I L'opération a abouti.

GSE0001E L'Extension Spatiale n'a pas pu effectuer l'opération demandée ("<nom-opération>") sous l'ID utilisateur ""<ID-utilisateur>".

Explications : Vous avez demandé une opération sous un ID utilisateur qui ne détient pas les droits nécessaires pour exécuter ladite opération.

Réponse de l'utilisateur : Consultez la documentation pour savoir quels sont les droits appropriés ou obtenez-les auprès de l'administrateur de DB2 Extension Spatiale.

GSE0002E "<valeur>" n'est pas une valeur autorisée pour l'argument "<nom-argument>".

Explications : La valeur que vous avez indiquée est incorrecte ou mal orthographiée.

Réponse de l'utilisateur : Consultez la documentation ou adressez-vous à un administrateur de DB2 Extension Spatiale pour connaître la valeur ou la plage de valeurs à indiquer.

GSE0003E L'extension spatiale n'a pas pu effectuer l'opération demandée car l'argument "<nom-argument>" n'est pas spécifié.

Explications : Vous n'avez pas spécifié un argument obligatoire pour cette opération.

Réponse de l'utilisateur : Spécifiez l'argument "<nom-argument>" avec la valeur de votre choix, puis demandez de nouveau l'opération.

GSE0004W L'argument "<nom-argument>" n'a pas été évalué.

Explications : L'opération demandée n'a pas utilisé l'argument "<nom-argument>".

Réponse de l'utilisateur : Aucune.

GSE0005E L'Extension Spatiale n'a pas pu traiter votre demande de création de l'objet "<nom-objet>".

Explications : L'objet "<nom-objet>" existe déjà ou vous n'avez pas les droits requis pour le créer. Il peut s'agir d'une table, d'une colonne, d'un déclencheur, d'un index, d'un fichier ou de tout autre type d'objet.

Réponse de l'utilisateur : Si "<nom-objet>" est l'objet que vous souhaitez créer, ne faites rien. Dans le cas contraire, spécifiez correctement le nom et vérifiez que vous disposez des droits requis pour créer cet objet.

GSE0006E L'Extension Spatiale n'a pas pu effectuer l'opération demandée sur un objet "<nom-objet>" activé ou enregistré.

Explications : L'objet "<nom-objet>" est déjà activé ou enregistré, ou il existe déjà. Il peut s'agir d'une couche, d'un index, d'un système de références spatiales, d'un système de coordonnées, d'un géocodeur ou de tout autre type d'objet.

Réponse de l'utilisateur : Vérifiez que l'objet "<nom-objet>" existe et soumettez de nouveau votre demande.

GSE0007E L'Extension Spatiale n'a pas pu effectuer l'opération demandée sur "<nom-objet>", objet qui n'a pas encore été activé ou enregistré.

Explications : L'objet "<nom-objet>" n'a pas été activé ou enregistré. Il peut s'agir d'une couche, d'un index, d'un système de références spatiales, d'un système de coordonnées spatiales, d'un géocodeur ou de tout autre type d'objet.

Réponse de l'utilisateur : Activez ou enregistrez l'objet "<nom-objet>" puis soumettez de nouveau votre demande.

GSE0008E Une erreur SQL ("<message-erreur-sql>") inattendue s'est produite.

Réponse de l'utilisateur : Consultez le message détaillé associé au SQLCODE du message d'erreur SQL "<message-erreur-sql>". Si nécessaire, prenez contact avec le service de maintenance IBM.

GSE0009E L'opération demandée n'a pas pu être exécutée sur un objet "<nom-objet>" qui existe déjà.

Explications : Il existe déjà un objet appelé "<nom-objet>" dans la base de données ou le système d'exploitation. Il peut s'agir d'un fichier, d'une table, d'une vue, d'une colonne, d'un déclencheur, d'un index ou de tout autre type d'objet.

Réponse de l'utilisateur : Vérifiez que vous spécifiez correctement l'objet lorsque vous tentez d'y accéder. Si nécessaire, supprimez-le.

GSE0010E L'opération demandée n'a pas pu être exécutée sur un objet "<nom-objet>" n'existe peut-être pas.

Explications : "<nom-objet>" n'existe pas dans la base de données ou le système d'exploitation. Il peut s'agir d'un fichier, d'une table, d'une vue, d'une colonne, d'un déclencheur, d'un index ou de tout autre type d'objet.

Réponse de l'utilisateur : Vérifiez que vous avez les droits nécessaires pour accéder à cet objet. Si tel est le cas et que l'objet n'existe pas, vous devez le créer.

GSE0011E L'Extension Spatiale n'a pas pu désactiver ou désenregistrer l'objet "<nom-objet>".

Explications : "<nom-objet>" est dépendant d'un autre objet. Il peut s'agir d'une couche, d'un index, d'un système de références spatiales, d'un géocodeur ou de tout autre type d'objet.

Réponse de l'utilisateur : Consultez la

documentation pour connaître les types d'objet dont "<nom-objet>" peut être dépendant. Supprimez ensuite l'objet spécifique dont "<nom-objet>" dépend.

GSE0012E L'Extension Spatiale n'a pas pu traiter votre demande car la colonne spatiale au nom qualifié complet "<schéma-couche.nom-couche.nom-colonne>" n'est pas enregistrée en tant que couche de table.

Explications : La colonne spatiale au nom qualifié complet "<schéma-couche.nom-couche.colonne-couche>" doit être enregistrée en tant que couche de table pour que vous puissiez effectuer certaines opérations la concernant (activation de l'index associé, activation d'un géocodeur pour la peupler en mode différé ou la mettre à jour automatiquement, etc.).

Réponse de l'utilisateur : Vérifiez que la colonne spatiale au nom qualifié complet "<schéma-couche.nom-couche.colonne-couche>" est enregistrée en tant que couche de table. Pour cela, consultez la vue DB2GSE.GEOMETRY_COLUMNS du catalogue Extension Spatiale. Assurez-vous également que la table contenant cette colonne comporte également des colonnes attribut correspondantes correctes.

GSE0013E La base de données n'est pas activée pour effectuer des opérations d'extension spatiale.

Explications : La base de données n'est pas activée pour effectuer des opérations d'extension spatiale. Par conséquent, le catalogue Extension Spatiale n'existe pas.

Réponse de l'utilisateur : Configurez la base de données pour les opérations spatiales.

GSE0014E La base de données a déjà été configurée pour les opérations spatiales.

Explications : La base de données a déjà été configurée pour les opérations spatiales.

Réponse de l'utilisateur : Vérifiez que la base de données a été activée comme vous le souhaitiez. Au besoin, désactivez-la.

GSE0498E L'erreur suivante s'est produite : "<message-erreur>".

GSE0499W L'Extension Spatiale a émis l'avertissement suivant : "<avertissement>".

GSE0500E Le mode de fonctionnement spécifié ("<mode-fonctionnement>") est incorrect.

Explications : Le mode spécifié n'est pas pris en charge par l'opération que vous avez demandée.

Réponse de l'utilisateur : Consultez la documentation pour connaître les modes pris en charge par cette opération.

GSE1001E L'Extension Spatials n'a pas pu enregistrer une couche de vue appelée "<nom-schéma.nom-vue.nom-colonne>" et basée sur la colonne spatiale "<nom-schéma.nom-table.nom-colonne>".

Explications : La colonne spatiale que vous avez spécifiée ("<nom-schéma.nom-table.nom-colonne>") n'a pas été enregistrée en tant que couche de table.

Réponse de l'utilisateur : Enregistrez la colonne "<nom-schéma.nom-table.nom-colonne>" en tant que couche de table.

GSE1002E L'Extension Spatiale n'a pas pu enregistrer une couche de vue appelée "<nom-schéma.nom-vue.nom-colonne>" et basée sur la table "<nom-schéma.nom-table>".

Explications : La table que vous avez indiquée ("<nom-schéma.nom-table>") n'est pas sous-jacente de la vue "<nom-schéma.nom-

vue.nom-colonne>", que ce soit directement ou indirectement.

Réponse de l'utilisateur : Déterminez la table qui est sous-jacente de la vue "<nom-schéma.nom-vue.nom-colonne>" et définissez-la.

GSE1003E **L'Extension Spatiale n'a pas pu accéder à la colonne intitulée "<nom-colonne>" dans la table ou la vue "<nom-schéma.nom-objet>".**

Explications : La table ou la vue "<nom-schéma.nom-objet>" ne comporte pas de colonne "<nom-colonne>".

Réponse de l'utilisateur : Vérifiez la définition de la table ou de la vue "<nom-schéma.nom-objet>" pour trouver le nom correct de la colonne souhaitée.

GSE1004E **L'Extension Spatiale n'a pas pu enregistrer la colonne spatiale au nom qualifié complet "<nom-schéma.nom-table.nom-colonne>" en tant que couche de table.**

Explications : La colonne "<nom-schéma.nom-table.nom-colonne>" n'est pas associée à un type de données spatiales ou à une table de base.

Réponse de l'utilisateur : Définissez un type de données spatiales pour la colonne "<nom-schéma.nom-table.nom-colonne>" ou assurez-vous qu'elle fait partie d'une table de base locale.

GSE1005E **Le système de références spatiales ("<ID-réf-spatiale-couche-vue>") que vous avez spécifié pour une couche de vue est différent de celui ("<ID-réf-spatiale-couche-table>") utilisé pour la couche de table sous-jacente correspondante.**

Explications : Le système de références spatiales d'une couche de vue doit être identique à celui de la couche de table sous-jacente.

Réponse de l'utilisateur : Spécifiez pour la couche de vue le système de références spatiales de la couche de table sous-jacente.

GSE1006E **L'ID système de références spatiales "<ID-références-spatiales>" est incorrect. L'Extension Spatiale n'a pas pu enregistrer la couche demandée.**

Explications : Le système de références spatiales que vous avez spécifié ("<ID-références-spatiales>") n'a pas été activé ou enregistré.

Réponse de l'utilisateur : Activez ou enregistrez le système de références spatiales, puis soumettez de nouveau la demande d'enregistrement de la couche.

GSE1007E **Une erreur SQL (SQLSTATE "<sqlstate>") s'est peut-être produite lorsque l'Extension Spatiale a tenté d'ajouter une colonne spatiale ("<nom-colonne>") à la table ("<nom-schéma.nom-table>".**

Réponse de l'utilisateur : Consultez le message associé au SQLSTATE "<sqlstate>".

GSE1008E **L'Extension Spatiale n'a pas pu enregistrer une couche de vue "<schéma-couche.nom-couche.colonne-couche>" car le type de données spatiales "<type-colonne-couche>" de la couche vue ne correspond pas au type de données spatiales "<type-colonne-géo>" de la couche de table sous-jacente "<schéma-géo.nom-géo.colonne-géo>".**

Explications : Le type de données spatiales d'une couche de vue "<schéma-couche.nom-couche.colonne-couche>" doit correspondre à celui de la couche de table sous-jacente de la dite couche de vue "<schéma-géo.nom-géo.colonne-géo>". L'incohérence entre les deux types de

données crée une ambiguïté lors du traitement des données spatiales.

Réponse de l'utilisateur : Veillez à ce que les types de données spatiales de la couche de vue et de la couche de table sous-jacente soient identiques.

GSE1020E **"<ID-références-spatiales>" est un ID système de références spatiales incorrect.**

Explications : Le système de références spatiales associé à l'identificateur "<ID-références-spatiales>" n'a pas été activé.

Réponse de l'utilisateur : Vérifiez que la référence spatiale spécifiée a été activée.

GSE1021E **L'Extension Spatiale n'a pas pu activer le système de références spatiales "<ID-références-spatiales>" car l'ID du système de coordonnées spatiales correspondant "<ID-coordonnées-spatiales>" est incorrect.**

Explications : Le système de coordonnées associé à l'identificateur "<ID-coordonnées-spatiales>" n'est pas défini dans le catalogue Extension Spatiale.

Réponse de l'utilisateur : Vérifiez l'identificateur de système de coordonnées "<ID-coordonnées-spatiales>" en consultant la vue DB2GSE.COORD_REF_SYS contenue dans le catalogue Extension Spatiale.

GSE1030E **"<nom-schéma.nom-table>" n'est pas une table de base ; l'Extension Spatiale n'a donc pas pu activer de géocodeur.**

Explications : L'objet contenant la source de données à géocoder doit être une table de base.

Réponse de l'utilisateur : Vérifiez que les colonnes contenant les données source à géocoder appartiennent à une table de base.

GSE1031E **L'Extension Spatiale n'a pas pu activer le géocodeur "<ID-géocodeur>" pour qu'il fonctionne automatiquement en mode Création pour la couche "<schéma-couche.nom-couche.colonne-couche>".**

Explications : Voici les explications possibles :

- Le géocodeur est déjà activé pour la mise à jour automatique de la couche "<schéma-couche.nom-couche.colonne-couche>".
- Le géocodeur a été temporairement invalidé pour cette couche.
- Aucune colonne de données source n'a été définie pour cette couche.

Réponse de l'utilisateur : Si le géocodeur a été temporairement invalidé, réactivez-le pour qu'il fonctionne automatiquement en mode Recréation.

GSE1032E **L'Extension Spatiale n'a pas pu activer le géocodeur "<ID-géocodeur>" pour qu'il fonctionne automatiquement en mode Recréation pour la couche "<schéma-couche.nom-couche.colonne-couche>".**

Explications : Voici les explications possibles :

- Le géocodeur est déjà activé pour la mise à jour automatique de la couche "<schéma-couche.nom-couche.colonne-couche>".
- Le géocodeur n'a pas été préalablement invalidé pour cette couche.
- Aucune colonne de données source n'a été définie pour cette couche.

Réponse de l'utilisateur : Si le géocodeur a été préalablement désactivé en mode Suppression ou qu'il n'a jamais été défini pour cette couche, activez-le pour qu'il fonctionne automatiquement en mode Création.

GSE1033E Une erreur SQL s'est produite lorsque l'Extension Spatiale a tenté d'ajouter des déclencheurs à la table qui contient la colonne destinée à la couche "<schéma-couche.nom-couche.colonne-couche>" (SQLSTATE "<sqlstate>").

Explications : Les déclencheurs servent à préserver l'intégrité des données entre les colonnes attribut dont proviennent les données en entrée du géocodeur et la colonne spatiale cible. L'erreur SQL s'est produite lorsque DB2 a tenté de créer ces déclencheurs.

Réponse de l'utilisateur : Consultez le message associé au SQLSTATE "<sqlstate>".

GSE1034E L'Extension Spatiale n'a pas pu désactiver le géocodeur "<ID-géocodeur>" en mode Suppression pour la couche "<schéma-couche.nom-couche.colonne-couche>".

Explications : Voici les explications possibles :

- Le géocodeur n'a jamais été activé pour la mise à jour automatique de la couche "<schéma-couche.nom-couche-colonne-couche>".
- Le géocodeur a été désactivé en mode Suppression.

Réponse de l'utilisateur : Déterminez l'état du géocodeur avant de tenter de le désactiver (s'il a été été enregistré, activé, etc.). Décidez ensuite s'il est nécessaire de le désactiver en mode Suppression. Ainsi, s'il n'a jamais été activé, il est inutile de le désactiver.

GSE1035E L'Extension Spatiale n'a pas pu désactiver le géocodeur "<ID-géocodeur>" en mode Invalidation pour la couche "<schéma-couche.nom-couche.colonne-couche>".

Explications : Voici les explications possibles :

- Le géocodeur n'a jamais été activé pour la mise à jour automatique de la couche "<schéma-couche.nom-couche-colonne-couche>".
- Le géocodeur a été désactivé en mode Invalidation ou en mode Suppression.

Réponse de l'utilisateur : Déterminez l'état du géocodeur avant de tenter de le désactiver (s'il a été été enregistré, activé, etc.). Décidez ensuite s'il est nécessaire de le désactiver en mode Invalidation. Ainsi, s'il a déjà été désactivé en mode Invalidation, il est inutile de répéter l'opération.

GSE1036E Une erreur SQL s'est produite lorsque l'Extension Spatiale a tenté de supprimer des déclencheurs de la table qui contient la colonne destinée à la couche "<schéma-couche.nom-couche.colonne-couche>" (SQLSTATE "<sqlstate>").

Explications : Les déclencheurs servent à préserver l'intégrité des données entre les colonnes attribut dont proviennent les données en entrée du géocodeur et la colonne spatiale cible. L'erreur SQL s'est produite lorsque DB2 a tenté de supprimer ces déclencheurs.

Réponse de l'utilisateur : Consultez le message associé au SQLSTATE "<sqlstate>".

GSE1037E L'Extension Spatiale n'a pas pu géocoder les données source de la couche de table "<schéma-couche.nom-couche.colonne-couche>", probablement en raison d'une valeur incorrecte "<nombre-attributs>" affectée à l'argument qui indique le nombre de colonnes attribut chargées de fournir des données source à cette couche.

Explications : Le nombre de colonnes attribut associées à cette couche ou le nom d'une ou de plusieurs colonnes est incorrect.

Réponse de l'utilisateur : Assurez-vous que la couche est enregistrée avec le nombre et les noms de colonnes attributs associées corrects, ou que les données d'entrée et de sortie du géocodeur sont également correctes.

GSE1038E **Une erreur SQL s'est produite lorsque l'Extension Spatiale a tenté de géocoder des données source pour la couche de table "<schéma-couche.nom-couche.colonne-couche>" en mode Traitement différé (SQLSTATE "<sqlstate>").**

Réponse de l'utilisateur :

- Consultez le message associé au SQLSTATE "<sqlstate>".
- Vérifiez que le contenu et l'argument UDF principal (primary UDF) de cette couche ont été définis correctement.

GSE1050E **La taille de grille spécifiée ("<taille-grille>") est incorrecte pour le premier niveau.**

Explications : Vous avez spécifié zéro ou un nombre négatif en tant que taille de grille pour le premier niveau.

Réponse de l'utilisateur : Spécifiez un nombre positif.

GSE1051E **La taille de grille spécifiée ("<taille-grille>") est incorrecte pour les deuxième et troisième niveaux.**

Explications : Vous avez spécifié un nombre négatif en tant que taille de grille pour le deuxième et le troisième niveaux.

Réponse de l'utilisateur : Spécifiez zéro ou un nombre positif.

GSE1052E **Une erreur SQL s'est produite lorsque l'Extension Spatiale a tenté de créer un index spatial "<schéma-index.colonne-index>" pour une couche de table "<schéma-couche.nom-couche.colonne-couche>" (SQLSTATE "<sqlstate>").**

Réponse de l'utilisateur :

- Vérifiez que l'index spatial est spécifié correctement et que la colonne spatiale n'est associée à aucun index.
- Consultez le message associé au SQLSTATE "<sqlstate>".

GSE1500I **Le géocodage de l'enregistrement source "<numéro-enregistrement>" a abouti.**

Explications : Le géocodage d'un enregistrement contenant des données d'attribut a abouti.

GSE1501W **L'enregistrement source "<numéro-enregistrement>" n'a pas été géocodé.**

Explications : Le degré de précision était trop élevé.

Réponse de l'utilisateur : Géocodez avec un degré de précision plus faible.

GSE1502W **L'enregistrement source "<numéro-enregistrement>" est introuvable.**

Réponse de l'utilisateur : Assurez-vous que l'enregistrement existe dans la base de données.

GSE2001E **Le fichier de transfert spécifié ("<nomfichier>") n'est pas correct.**

Réponse de l'utilisateur : Vérifiez que le fichier spécifié est un fichier de transfert SDE et que le chemin indiqué est correct.

GSE2002E La clause SQL WHERE fournie (“<clause-WHERE-SQL>”) n’est pas correcte.

Réponse de l’utilisateur : Vérifiez s’il n’y a pas d’erreur de syntaxe, de faute d’orthographe et de nom de colonne incorrect dans la clause WHERE.

GSE2003E La valeur de forme fournie n’est pas autorisée.

Réponse de l’utilisateur : Assurez-vous que la forme fournie correspond au type associé à la colonne spatiale. Si les types sont identiques ou compatibles, la forme géométrique n’est pas autorisée. Recherchez les polygones qui se chevauchent, les arcs formés d’un seul point, etc.

GSE2004E Le schéma du fichier de transfert est incompatible avec celui de la couche spécifiée.

Réponse de l’utilisateur : Vérifiez que les noms de schéma et de couche sont corrects. Si les schémas ne sont pas identiques, chargez les données dans une nouvelle table et résolvez les disparités existant entre les schémas.

GSE2005E La géométrie du fichier de transfert est incompatible avec celle de la couche spécifiée.

Réponse de l’utilisateur : Vérifiez que les noms de schéma et de couche sont corrects.

GSE2006E Une erreur d’E-S s’est produite pour un fichier intitulé “<nomfichier>”.

Réponse de l’utilisateur : Assurez-vous que le fichier existe et que vous disposez des droits d’accès appropriés, puis relancez l’opération.

GSE2007E Une erreur de conversion des attributs s’est produite.

Réponse de l’utilisateur : Assurez-vous que tous les types d’attribut de la table sont pris en charge. Par exemple, les données de type BLOB ne sont pas acceptées dans les fichiers de forme. Recherchez également s’il existe des valeurs de données n’appartenant pas à la plage définie ou non autorisées (format de date erroné, etc.).

GSE2008E La fonction d’importation/exportation ne dispose plus de mémoire.

Réponse de l’utilisateur : Vérifiez que vous disposez d’une quantité de mémoire appropriée.

Chapitre 11. Vues du catalogue

Les vues du catalogue DB2 Extension Spatiale contiennent des métadonnées concernant les ressources et objets suivants :

- Systèmes de coordonnées utilisables. Pour plus d'informations sur les identificateurs et les textes d'annotation concernant ces systèmes, reportez-vous à la section «DB2GSE.COORD_REF_SYS».
- Colonnes spatiales enregistrées en tant que couches. Pour plus d'informations sur les noms et types de données de ces colonnes, ainsi que sur les systèmes de références spatiales qui leur sont associés, reportez-vous à la section «DB2GSE.GEOMETRY_COLUMNS» à la page 116.
- Géocodeurs utilisables. Pour plus d'informations sur les identificateurs de ces géocodeurs et les régions contenant les lieux qu'ils traitent, reportez-vous à la section «DB2GSE.SPATIAL_GEOCODER» à la page 117.
- Systèmes de références spatiales utilisables. Pour plus d'informations sur les identificateurs et descriptions de ces systèmes, reportez-vous à la section «DB2GSE.SPATIAL_REF_SYS» à la page 117.

DB2GSE.COORD_REF_SYS

Lorsque vous activez une base de données pour des opérations spatiales, DB2 Extension Spatiale enregistre les systèmes de coordonnées que vous pouvez utiliser dans une table de catalogue. Les colonnes sélectionnées à partir de cette table forment la vue DB2GSE.COORD_REF_SYS du catalogue, qui est décrite dans le tableau 34.

Tableau 34. Colonnes de la vue DB2GSE.COORD_REF_SYS du catalogue

Nom	Type de données	Valeur NULL admise ?	Contenu
SCID	INTEGER	Non	Identificateur numérique unique associé à ce système de coordonnées.
SC_NAME	VARCHAR(64)	Non	Nom de ce système de coordonnées.
AUTH_NAME	VARCHAR(256)	Oui	Nom de l'organisme auquel adhère la société qui a compilé ce système de coordonnées ; par exemple, European Petroleum Survey Group (EPSG).
AUTH_SRID	INTEGER	Oui	Identificateur numérique affecté à ce système de coordonnées par l'organisme spécifié dans la colonne AUTH_NAME.

Tableau 34. Colonnes de la vue DB2GSE.COORD_REF_SYS du catalogue (suite)

Nom	Type de données	Valeur NULL admise ?	Contenu
DESC	VARCHAR(256)	Oui	Description de ce système de coordonnées.
SRTEXT	VARCHAR(2048)	Non	Texte d'annotation associé à ce système de coordonnées.

DB2GSE.GEOMETRY_COLUMNS

Lorsque vous créez une couche, DB2 Extension Spatiale l'enregistre en stockant son identificateur ainsi que des informations la concernant dans une table de catalogue. Les colonnes sélectionnées à partir de cette table forment la vue DB2GSE.GEOMETRY_COLUMNS_SYS du catalogue, qui est décrite dans le tableau 35.

Tableau 35. Colonnes de la vue DB2GSE.GEOMETRY_COLUMNS du catalogue

Nom	Type de données	Valeur NULL admise ?	Contenu
LAYER_CATALOG	VARCHAR(30)	Oui	Nom qualifié complet de cette couche.
LAYER_SCHEMA	VARCHAR(30)	Non	Schéma de la table ou de la vue contenant la colonne enregistrée en tant que cette couche.
LAYER_NAME	VARCHAR(128)	Non	Nom de la table ou de la vue contenant la colonne enregistrée en tant que cette couche.
LAYER_COLUMN	VARCHAR(30)	Non	Nom de la colonne enregistrée en tant que cette couche.
GEOMETRY_TYPE	INTEGER	Non	Type de données de la colonne enregistrée en tant que cette couche.
SRID	INTEGER	Non	Identificateur du système de références spatiales utilisé pour les valeurs contenues dans la colonne enregistrée en tant que cette couche.
STORAGE_TYPE	INTEGER	Oui	Informations sur le mode de stockage DB2 utilisé pour les valeurs contenues dans la colonne enregistrée en tant que couche. Par exemple, des données de type STORAGE_TYPE peuvent indiquer que les valeurs sont stockées en tant qu'objets de type LOB ou en tant qu'instances de types de données abstraits (ADT).

DB2GSE.SPATIAL_GEOCODER

Les géocodeurs disponibles sont enregistrés dans une table du catalogue. Les colonnes sélectionnées à partir de cette table forment la vue DB2GSE.SPATIAL_GEOCODER du catalogue, qui est décrite dans le tableau 36.

Tableau 36. Colonnes de la vue DB2GSE.SPATIAL_GEOCODER du catalogue

Nom	Type de données	Valeur NULL admise ?	Contenu
GCID	INTEGER	Non	Identificateur numérique de ce géocodeur.
GC_NAME	VARCHAR(64)	Non	Description succincte de ce géocodeur.
VENDOR_NAME	VARCHAR(128)	Non	Nom du fournisseur de ce géocodeur.
PRIMARY_UDF	VARCHAR(256)	Non	Nom qualifié complet de ce géocodeur.
PRECISION_LEVEL	INTEGER	Non	Degré de correspondance requis entre les données source et les données de référence pour que le géocodeur traite les données source avec succès.
VENDOR_SPECIFIC	VARCHAR(256)	Oui	Chemin d'accès et nom d'un fichier utilisable par le fournisseur pour définir tous les paramètres spéciaux pris en charge par ce géocodeur.
GEO_AREA	VARCHAR(256)	Oui	Zone géographique contenant les emplacements à géocoder.
DESCRIPTION	VARCHAR(256)	Oui	Remarques provenant du fournisseur.

DB2GSE.SPATIAL_REF_SYS

Lorsque vous créez un système de références spatiales, DB2 Extension Spatiale l'enregistre en stockant son identificateur ainsi que des informations le concernant dans une table de catalogue. Les colonnes sélectionnées à partir de cette table forment la vue DB2GSE.SPATIAL_REF_SYS du catalogue, qui est décrite dans le tableau 37.

Tableau 37. Colonnes de la vue DB2GSE.SPATIAL_REF_SYS

Nom	Type de données	Valeur NULL admise ?	Contenu
SRID	INTEGER	Non	Identificateur défini par l'utilisateur et associé au système de références spatiales.
SR_NAME	VARCHAR(64)	Non	Nom du système de références spatiales.

Tableau 37. Colonnes de la vue DB2GSE.SPATIAL_REF_SYS (suite)

Nom	Type de données	Valeur NULL admise ?	Contenu
SCID	INTEGER	Non	Identificateur numérique du système de coordonnées sous-jacent de ce système de références spatiales.
SC_NAME	VARCHAR(64)	Non	Nom du système de coordonnées sous-jacent de ce système de références spatiales.
AUTH_NAME	VARCHAR(256)	Oui	Nom de l'organisme qui définit les normes de ce système de références spatiales.
AUTH_SRID	INTEGER	Oui	Identificateur que l'organisme spécifié dans la colonne AUTH_NAME affecte à ce système de références spatiales.
SRTEXT	VARCHAR(2048)	Non	Texte d'annotation associé à ce système de références spatiales.

Chapitre 12. Index spatiaux

Les colonnes spatiales contiennent des données géographiques bidimensionnelles ; les applications qui les analysent exigent donc une stratégie d'indexation qui permette d'identifier rapidement toutes les géométries qui se trouvent dans un domaine donné. C'est pour cette raison que DB2 Extension Spatiale fournit un index spatial à trois niveaux fondé sur une grille.

Le présent chapitre décrit ce type d'index et fournit les instructions nécessaires à son utilisation. Il traite des sujets suivants :

Fragment du programme exemple

L'exemple ci-après illustre le processus de création et d'utilisation d'un index avec SQL. Pour plus d'informations sur les commandes CREATE INDEX et CREATE INDEX EXTENSION, reportez-vous au manuel *SQL Reference*. Vous remarquerez qu'une fois l'index créé, vous pouvez émettre des instructions DDL et DML classiques qui utilisent des fonctions et prédicats spatiaux.

```
rev="ver7a">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)
```

Index en arbre B

La technique d'indexation spatiale est fondée sur l'index hiérarchisé en arbre B classique, mais s'avère très différente de celui-ci. L'index spatial utilise une *indexation de type grille* conçue pour indexer des colonnes spatiales à deux dimensions. L'index en arbre B ne peut traiter que des données unidimensionnelles et ne peut donc pas être utilisé avec les informations SIG. La présente section décrit la structure et l'utilisation d'un index en arbre B.

Le niveau supérieur d'un index en arbre B, appelé noeud racine, contient une clé pour chaque noeud appartenant au niveau suivant. La valeur de chaque clé est la valeur de clé existante la plus importante associée au noeud correspondant dans le niveau suivant. En fonction du nombre de valeurs contenues dans la table de base, plusieurs noeuds intermédiaires peuvent s'avérer nécessaires. Ils forment un pont entre le noeud racine et les noeuds feuille qui contiennent les ID des lignes de la table de base.

Le gestionnaire de bases de données analyse un index en arbre B en commençant par le noeud racine, puis en continuant par les noeuds intermédiaires jusqu'il atteigne le noeud feuille contenant l'ID de ligne de la table de base.

L'index en arbre B est incompatible avec une colonne spatiale car la nature bidimensionnelle de ce type de colonne exige une structure d'index spatial. Pour cette même raison, vous ne pouvez pas utiliser un index spatial pour une colonne non spatiale, et vous ne pouvez pas non plus l'appliquer à une colonne composée de quelque type que ce soit.

Modes de création d'un index spatial

Il existe plusieurs modes de création d'un index spatial :

- Définition à partir de la fenêtre Création d'un index spatial. Pour la procédure correspondante, reportez-vous au «Chapitre 6. Création d'index spatiaux» à la page 53.
- Appel de la procédure mémorisée db2gse.gse_enable_idx dans un programme d'application. Pour plus d'informations sur cette procédure, reportez-vous au «Chapitre 9. Procédures mémorisées» à la page 69.

- Emission de la commande **db2 create index** comportant la fonction **spatial_index** dans la clause USING. Par exemple :

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

La nature des données spatiales implique que le concepteur de la base de données comprenne leur distribution relative en termes de volume. Il doit déterminer la taille et le nombre optimums de niveaux de grille avec lesquels créer l'index spatial.

Les niveaux de grille, <niveau de grille 1>, <niveau de grille 2> et <niveau de grille 3>, sont définis en augmentant la taille de la cellule. Ainsi, le niveau intermédiaire doit avoir une cellule d'une taille supérieure au premier niveau, et le troisième, une cellule d'une taille supérieure au second. Le premier niveau de grille est obligatoire mais vous pouvez désactiver le second et le troisième avec une valeur zéro à double précision (0.0e0).

Mode de génération d'un index spatial

Un index spatial est généré à l'aide d'*enveloppes*. L'enveloppe est elle-même une géométrie ; elle représente les dimensions X et Y minimales et maximales d'une géométrie. Dans la plupart des cas, l'enveloppe est une boîte mais pour les lignes horizontales et verticales, il s'agit d'une ligne passant par deux points. Pour les points, l'enveloppe est réduite au point même. Pour plus d'informations sur les enveloppes, reportez-vous à la section «Enveloppe» à la page 134.

L'index spatial est construit sur une colonne spatiale en créant une ou plusieurs entrées pour les intersections de l'enveloppe de chaque géométrie avec la grille. Une intersection est enregistrée sous la forme de l'ID interne de la figure et des coordonnées X et Y de la cellule de grille intersectée. Par exemple, le polygone de la figure 7 à la page 122, forme une intersection avec la grille aux coordonnées (20,30), (30,30), (40,30), (20,40), (30,40), (40,40), (20,50), (30,50) et (40,50). Reportez-vous au tableau 38 à la page 123, pour les abscisses et ordonnées (coordonnées X et Y) minimales de toutes les géométries de la figure 7 à la page 122.

En présence de plusieurs niveaux de grille, DB2 Extension Spatiale tente d'utiliser le niveau le plus faible possible. Lorsque la géométrie forme une intersection avec au moins quatre cellules de grille à un niveau donné, elle est promue au niveau immédiatement supérieur. Par conséquent, pour un index spatial doté de trois niveaux de grille, 10.0e0, 100.0e0 et 1000.0e0, DB2 Extension Spatiale génère tout d'abord l'intersection de chaque géométrie avec la grille de niveau 10.0e0. Si une géométrie forme une intersection avec au moins quatre cellules de la grille 10.0e0, elle est promue et croisée avec la grille de niveau 100.0e0. Si l'on obtient au moins quatre intersections au

niveau 100.0e0, elle est alors promue au niveau 1000.0e0. A ce stade, les intersections doivent être enregistrées dans l'index spatial car il s'agit du niveau le plus élevé possible.

La figure 7, illustre comment quatre types de géométries différents forment des intersections avec une grille de niveau 10.0e. Les 23 intersections associées à ces quatre géométries sont enregistrées dans l'index spatial.

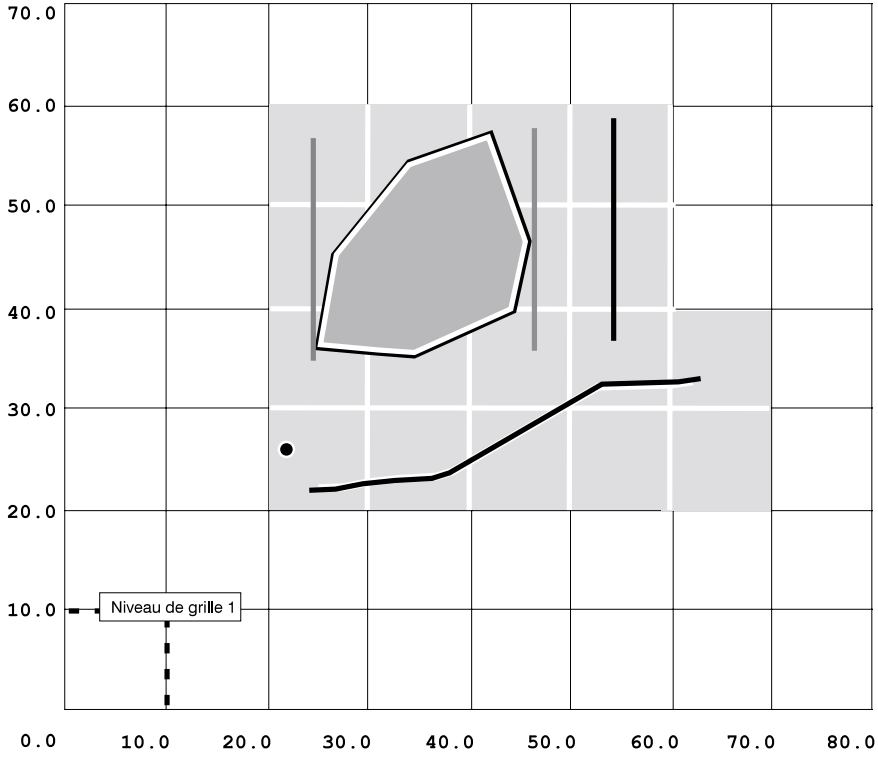


Figure 7. Application d'un niveau de trame de 10.0e0

Le tableau 38 à la page 123 répertorie les géométries et leurs intersections avec la grille. Les enveloppes de quatre types de géométries différents forment des intersections avec la grille de niveau 10.0e. L'abscisse (X) et l'ordonnée (Y) minimales de chaque cellule de la trame qu'elle coupe sont enregistrées dans l'index spatial.

Tableau 38. Entrées des cellules de grille au 10.0e0 associées aux géométries exemples

Géométrie	X de la grille	Y de la grille
Polygone	20.0	30.0
Polygone	30.0	30.0
Polygone	40.0	30.0
Polygone	20.0	40.0
Polygone	30.0	40.0
Polygone	40.0	40.0
Polygone	20.0	50.0
Polygone	30.0	50.0
Polygone	40.0	50.0
Ligne verticale	50.0	30.0
Ligne verticale	50.0	40.0
Ligne verticale	50.0	50.0
Point	20.0	20.0
Ligne horizontale	20.0	20.0
Ligne horizontale	30.0	20.0
Ligne horizontale	40.0	20.0
Ligne horizontale	50.0	20.0
Ligne horizontale	60.0	20.0
Ligne horizontale	20.0	30.0
Ligne horizontale	30.0	30.0
Ligne horizontale	40.0	30.0
Ligne horizontale	50.0	30.0
Ligne horizontale	60.0	30.0

La figure 8 à la page 124, montre comment le nombre d'intersections est considérablement réduit à 8 en ajoutant les niveaux de grille 30.0e0 et 60.0e0. Dans ce cas, le polygone identifié en tant que figure géométrique 1 est promu au niveau de grille 30.0e0 et la ligne identifiée en tant que figure 4 est promue au niveau de grille 60.0e0. Au lieu des neuf et dix intersections respectives existant au niveau 10.0e0, ces deux géométries n'en comptent plus que deux après leur promotion.

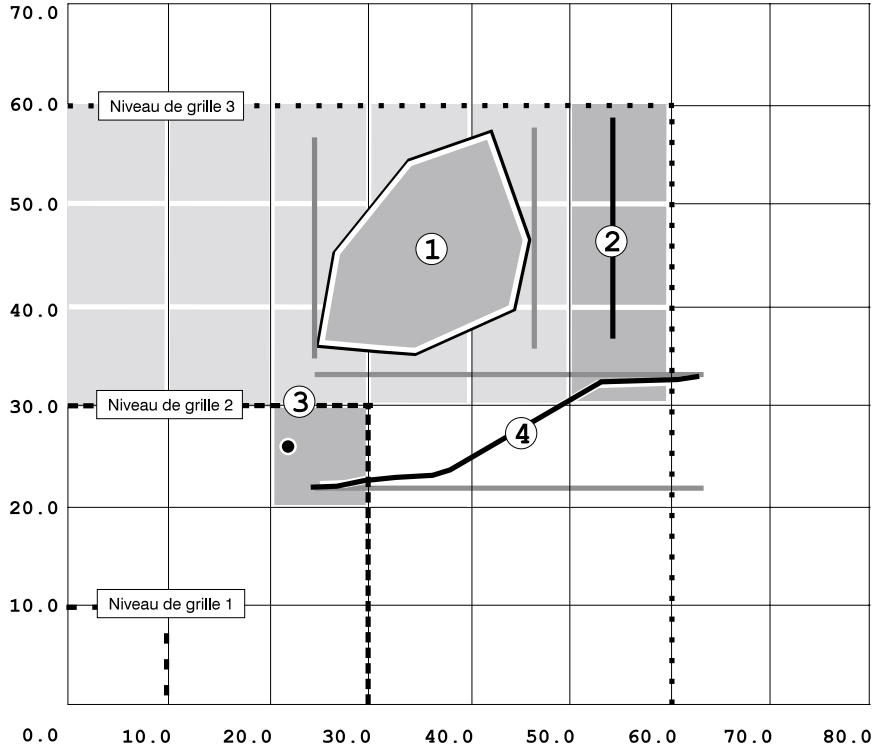


Figure 8. Incidence de l'ajout des niveaux de grille 30.0e0 et 60.0e0. L'enveloppe du polygone identifié en tant que géométrie 1 génère des intersections avec neuf cellules de la grille. Celle de la ligne verticale (géométrie 2) coupe trois cellules de la grille. L'enveloppe du point (géométrie 3) ne génère une intersection qu'avec une seule cellule et celle de la ligne identifiée en tant que géométrie 4 coupe dix cellules de la grille.

DB2 Extension Spatiale vérifie chaque objet spatial par rapport aux paramètres de niveau de grille spécifié dans l'instruction CREATEINDEX, afin de déterminer le nombre et les coordonnées des blocs de grille dans lequel l'objet se trouve. Dans la figure 8, les niveaux de grille 10.0e0, 30.0e0 et 60.0e0 sont représentés par des traits de lignes de plus en plus appuyés et des nuances de gris différentes. Les intersections de la ligne verticale et de l'enveloppe du point avec les cellules sont enregistrées dans l'index au niveau de grille 10.0e0, car ces deux géométries génèrent moins de quatre intersections. Le polygone forme des intersections avec neuf cellules de la grille 10.0e0 et il est donc promu au niveau de grille 30.0e0. A ce niveau, il coupe deux cellules de la grille, qui sont entrées dans l'index. La ligne identifiée en tant que géométrie 4 forme des intersections avec 10 cellules de grille 10.0e0 et elle est donc promue au niveau de grille 30.0e0. Mais même à ce niveau, elle coupe encore six cellules de grille et elle est donc à nouveau promue, cette fois au niveau de grille 60.0e0 où elle génère deux intersections. Les intersections de la ligne avec la grille 60.0e0 sont alors entrées dans l'index. Si cette ligne avait généré quatre intersections ou plus à ce niveau, celles-ci auraient tout de

même été enregistrées dans l'index car il s'agit du niveau le plus élevé de promotion d'une géométrie.

Tableau 39. Intersections des géométries dans l'index à trois niveaux

Géométrie	X de la grille	Y de la grille
Intersections de la ligne verticale et du point au niveau 1 (trame de grille 10.0e0)		
2	50.0	30.0
2	50.0	40.0
2	50.0	50.0
3	20.0	20.0
Intersections du polygone au niveau 2 (trame de grille 30.0e0)		
1	0.0	30.0
1	30.0	30.0
Intersections de la ligne au niveau 3 (trame de grille 60.0e0)		
4	0.0	0.0
4	60.0	0.0

En fait, DB2 Extension Spatiale ne crée pas réellement de structure de grille polygonale. Il représente chaque niveau de grille sous forme de paramètres en définissant l'origine du décalage des abscisses (X) et des ordonnées (Y) du système de références spatiales des colonnes. Il agrandit ensuite la grille en un espace de coordonnées positives. DB2 Extension Spatiale génère mathématiquement les intersections à l'aide d'une grille paramétrique.

Utilisation des index spatiaux

DB2 Extension Spatiale recourt à un index spatial pour améliorer les performances d'une analyse spatiale. Prenons l'analyse spatiale la plus élémentaire et probablement la plus utilisée : l'analyse par boîte. Elle consiste à demander à DB2 Extension Spatiale de renvoyer toutes les géométries qui sont contenues partiellement ou en totalité dans une boîte définie par l'utilisateur. Si aucun index n'existe, DB2 Extension Spatiale doit comparer toutes les géométries à la boîte. Par contre, avec un index, DB2 Extension Spatiale peut localiser toutes les entrées d'index dont la coordonnée gauche la plus basse est supérieure ou égale à celle de la boîte et dont la coordonnée droite la plus haute est plus petite ou égale à celle de la boîte. L'index étant classé d'après ce système de coordonnées, DB2 Extension Spatiale peut obtenir rapidement la liste des géométries susceptibles de répondre aux conditions requises. Il s'agit du processus appelé *première passe*.

Une *seconde passe* détermine si l'enveloppe de chacune des géométries présélectionnées génère des intersections avec la boîte. En effet, une géométrie qui remplit les conditions de la première passe parce que l'enveloppe associée à ses cellules de grille forme des intersections avec la boîte, peut avoir elle-même une enveloppe qui n'en forme pas.

Une *troisième passe* compare les coordonnées réelles de la géométrie présélectionnée avec la boîte afin de déterminer si une partie quelconque de cette géométrie se trouve réellement dans la boîte. Ce dernier processus de comparaison plutôt complexe est exécuté sur une liste de géométries présélectionnées composée d'un sous-ensemble de la population totale, considérablement réduite par les deux premières passes.

Toutes les analyses spatiales exécutent ces trois passes, sauf la fonction EnvelopesIntersect. Celle-ci n'effectue que les deux premières passes. En effet, la fonction EnvelopesIntersect a été conçue pour les opérations d'affichage qui utilisent souvent leurs propres routine intégrées de découpage et ne requièrent pas la granularité de la troisième passe.

Sélection de la taille des cellules de la grille

La forme irrégulière des enveloppes des géométries complique la sélection de la taille des cellules de la grille. En effet, certaines enveloppes forment des intersections avec plusieurs grilles alors que d'autres sont entièrement contenues dans une seule cellule de grille. Inversement, en fonction de la répartition spatiale des données, certaines cellules de grille peuvent générer des intersections avec de nombreuses enveloppes de géométries.

Pour qu'un index spatial remplisse bien sa fonction, il est essentiel de sélectionner correctement le nombre et la trame des grilles. Prenons une colonne spatiale contenant des géométries de taille uniforme. Dans ce cas, il suffit d'un seul niveau de grille. Choisissez tout d'abord une taille de cellule de grille qui contienne l'enveloppe de géométrie moyenne. En testant votre application, vous constaterez éventuellement qu'en augmentant la taille de la cellule, vous améliorez les performances de vos requêtes. Cela s'explique par le fait que chaque cellule de grille contient plusieurs géométries et que la première passe a permis d'écarter plus rapidement celles qui ne remplissaient pas les conditions requises. Toutefois, vous constaterez que les performances se dégradent à mesure que vous continuez à augmenter la taille de la cellule. Cela est dû au fait que la seconde passe devra éventuellement traiter plus de candidats.

Sélection du nombre de niveaux

Si les objets à index sont d'une taille relative à peu près identique, vous pouvez utiliser un seul niveau de grille. Bien que cela soit vrai, toutes les colonnes ne contiennent pas des géométries de même taille relative. En règle générale, les géométries des colonnes spatiales peuvent être regroupées en

plusieurs plages de tailles. Prenons l'exemple d'un réseau routier dans lequel les géométries sont réparties en rues, routes principales et autoroutes. Les rues sont toutes à peu près de la même longueur et peuvent être regroupées dans une même plage de tailles. Il en va de même pour les routes principales et les autoroutes. Par conséquent, les rues représentant une plage de taille peuvent être regroupées dans un premier niveau de grille, les routes principales dans un second et les autoroutes dans un troisième. Un autre exemple comprend une colonne parcelles d'un comté comportant des groupes de petites parcelles urbaines entourées de parcelles rurales de grande taille. Dans ce cas, il existe deux plages de taille et deux niveaux de grille, un pour les petites parcelles urbaines et l'autre pour les grandes parcelles rurales. Cela arrive fréquemment et nécessite l'utilisation d'une grille à plusieurs niveaux.

Pour sélectionner la taille de cellule associée à chaque niveau de grille, sélectionnez des tailles légèrement supérieures à chaque plage de tailles. Testez ensuite l'index en exécutant des analyses sur la colonne spatiale.

Chaque niveau supplémentaire exige un nouveau balayage de l'index. Diminuez ou augmentez légèrement les tailles des grilles pour déterminer si on peut obtenir un gain de performances sensible.

Chapitre 13. Géométries et fonctions spatiales associées

Le présent chapitre décrit les unités d'informations appelées *géométries* qui consistent en des coordonnées et symbolisent des entités géographiques. Il présente également les fonctions spatiales qui utilisent ces figures en tant qu'entrée et renvoient des résultats permettant d'analyser des entités géographiques et de transférer des données spatiales entres systèmes d'information géographique. Il traite des sujets suivants :

- Nature des géométries
- Propriétés des géométries, fonctions renvoyant des informations relatives à ces propriétés
- Géométries instanciables, fonctions les utilisant
- Fonctions permettant :
  - de visualiser des relations et des comparaisons entre entités géographiques
  - de générer des géométries
  - de convertir des valeurs de géométrie dans un format importable et exportable

Présentation des géométries

Le dictionnaire Oxford American Dictionary définit le terme *geometry* comme «la branche des mathématiques traitant des propriétés et des relations des lignes, angles, surfaces et solides.» Le 11 août 1997, le groupement Open GIS Consortium Inc. (OGC) dans sa publication *Open GIS Features for ODBC (SQL) Implementation Specification*, a ajouté une autre définition de ce terme. Le mot *géométrie* a été choisi pour désigner les figures géométriques qui, depuis un millénaire ou plus, ont été utilisées par les cartographes pour tracer les cartes du monde. Une définition très abstraite de cette nouvelle signification du terme géométrie pourrait être la suivante : «point ou agrégat de points symbolisant une entité sur le sol.»

Dans DB2 Extension Spatiale, une définition *opérationnelle* du terme géométrie pourrait être «modélisation d'une entité géographique». Le modèle peut s'exprimer sous la forme des coordonnées de l'entité et également, dans certains cas, sous la forme d'un symbole visuel. Le modèle véhicule des informations ; ainsi, les coordonnées identifient la position de l'entité par rapport à des points de référence fixes et le symbole délimite sa forme. En outre, le modèle peut servir à générer des informations ; par exemple, la

fonction ST_Overlaps utilise les coordonnées de deux régions voisines en tant qu'entrée et renvoie des informations indiquant si ces deux régions se chevauchent.

Les coordonnées d'une entité symbolisée par une géométrie sont considérés comme les *propriétés* de la géométrie. Différents types de géométrie ont également d'autres propriétés ; par exemple :

- L'*intérieur* représente le contenu de l'entité symbolisée par la géométrie.
- L'*extérieur* représente l'espace situé autour de l'entité.
- Le *contour* représente la démarcation entre la fin du contenu d'une géométrie et le début de son espace d'entourage.

Ces propriétés ainsi que d'autres sont traitées à la section «Propriétés des géométries et fonctions associées» à la page 131.

Les géométries prises en charge par DB2 Extension Spatiale forment une structure hiérarchique, illustrée à la figure 9. Six de ses membres sont instanciables ; autrement dit, il peuvent être représentés en tant que symboles visuels, également illustrés dans cette figure.

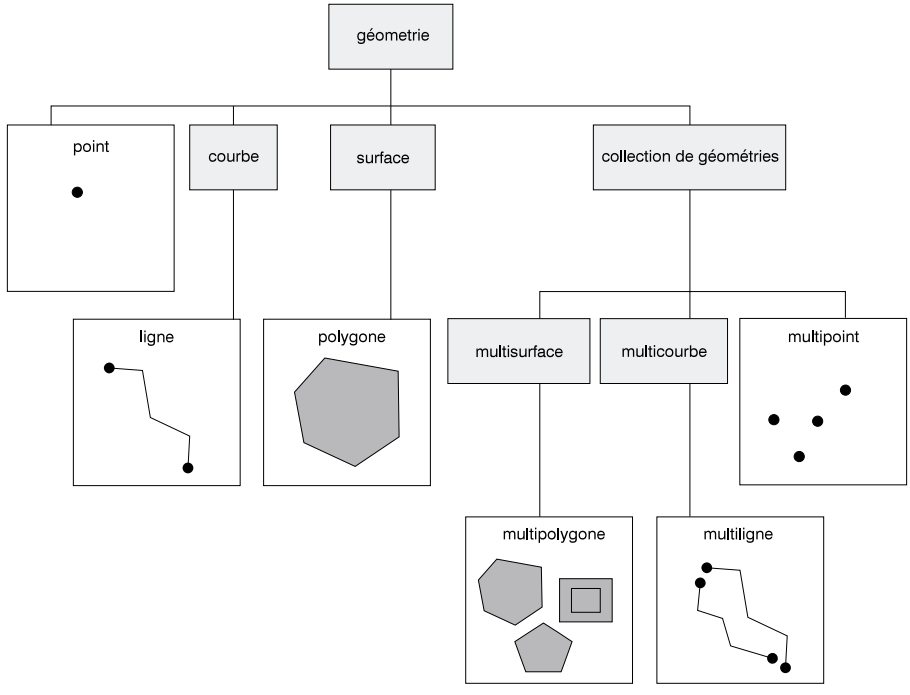


Figure 9. Structure hiérarchique des géométries prises en charge par DB2 Extension Spatiale. Les géométries instanciables peuvent être exprimées sous forme de symboles visuels. Ceux-ci sont représentés sous le nom de la géométrie correspondante.

Comme illustré à la figure 9 à la page 130, une superclasse appelée *géométrie* se situe au sommet (root) de la hiérarchie. Les sous-classes sont réparties en deux catégories : les sous-classes de géométries élémentaires et les sous-classes de collections homogènes. Les géométries élémentaires comprennent :

- les *points*, qui symbolisent des entités discrètes perçues comme occupant le lieu géométrique où une ligne de coordonnées est-ouest (parallèle, etc) coupe une ligne de coordonnées nord-sud (méridien, etc). Par exemple, supposons que le système de notation sur une carte à grande échelle indique que chaque ville figurant sur la carte est localisée à l'intersection d'un parallèle et d'un méridien. A cette échelle, chaque ville serait symbolisé par un point.
- les *lignes*, qui symbolisent des entités géographiques linéaires (rues, canaux, pipelines, etc.).
- les *polygones*, qui symbolisent des entités géographiques dotées de plusieurs côtés (forêts, réserves naturelles, etc.).

Les collections homogènes regroupent :

- les *multipoints*, qui symbolisent des entités composées dont chaque élément est situé à l'intersection d'une ligne de coordonnées est-ouest et d'une ligne de coordonnées nord-sud (par exemple, un archipel dont chaque île est située à l'intersection d'un parallèle et d'un méridien).
- les *multilignes*, qui symbolisent des entités composées constituées d'unités ou d'éléments linéaires (systèmes hydrographiques, autoroutiers, etc.).
- les *multipolygones*, qui symbolisent des entités composées constituées d'unités ou d'éléments dotés de plusieurs côtés (zones agricoles collectives dans une région déterminée, système de lacs, etc.).

Comme leur nom l'indique, les collections homogènes sont des ensembles de géométries élémentaires. Outre le fait de partager les propriétés de ces géométries, les collections homogènes en possèdent d'autres qui leur sont propres.

Les données spatiales prise en charge par DB2 Extension Spatiale sont des implémentations des géométries illustrées à la figure 9 à la page 130. Pour la description de ces types de données, reportez-vous à la section «Présentation des types de données spatiales» à la page 33.

Propriétés des géométries et fonctions associées

La présente section décrit les propriétés des géométries et les fonctions spatiales associées à ces dernières. Elle commence par les principales propriétés :

- Classe à laquelle appartient une géométrie
- Abscisses (X) et ordonnées (Y)

ainsi que :

- Coordonnées Z
- Mesures
- Intérieur, extérieur et contour d'une géométrie
- Simple ou complexe
- Vide ou non vide
- Enveloppe d'une géométrie
- Dimension
- Identificateur du système de références spatiales associé à une géométrie

Classe

Chaque géométrie appartient à une classe de la hiérarchie illustrée à la figure 9 à la page 130. Comme indiqué à la section «Présentation des géométries» à la page 129, six sous-classes (points, lignes, polygones, multipoints, multilignes et multipolygones) sont instanciables. La superclasse et les autres sous-classes ne le sont pas.

La fonction ST_GeometryType utilise une géométrie en entrée et renvoie la sous-classe instanciable sous la forme d'une chaîne de caractères. Pour plus d'informations, reportez-vous à la section «ST_GeometryType» à la page 227.

La fonction ST_IsValid utilise en entrée une géométrie affectée à un type de données ST_Geometry. Elle renvoie la valeur 1 (TRUE) si la géométrie est valide et 0 (FALSE) dans le cas contraire. Pour plus d'informations, reportez-vous à la section «ST_IsValid» à la page 245.

Coordonnées X et Y

Une valeur d'*abscisse (X)* désigne un emplacement défini par rapport à un point de référence à l'est ou à l'ouest. Une valeur d'*ordonnée (Y)* désigne un emplacement défini par rapport à un point de référence au nord ou au sud. Pour plus d'informations, reportez-vous aux sections «Nature des données spatiales» à la page 6, et «Présentation des systèmes de références spatiales et de coordonnées» à la page 25.

Coordonnées Z

Certaines géométries sont associées à une altitude ou à une profondeur. Chaque point constituant la géométrie d'une entité peut comprendre une coordonnée Z facultative qui représente la normale d'altitude ou de profondeur par rapport à la surface de la terre.

Le prédicat Is3d utilise une géométrie en entrée et renvoie la valeur 1 (TRUE) si la fonction a des coordonnées Z et 0 (FALSE), dans le cas contraire. Pour plus d'informations, reportez-vous à la section «Is3d» à la page 173.

Mesures

Une mesure est une valeur qui véhicule des informations sur une entité géographique et qui est stockée avec les coordonnées définissant l'emplacement de la dite entité. Par exemple, supposons que vous représentiez des systèmes de transport sur votre SIG. Pour que votre application traite les valeurs désignant des distances linéaires ou des poteaux kilométriques, vous pouvez stocker ces valeurs avec les coordonnées définissant l'emplacement des systèmes. Les mesures sont enregistrées sous la forme de nombres à double précision.

Le prédicat IsMeasured utilise une géométrie en entrée et renvoie la valeur 1 (TRUE) si elle contient des mesures et 0 (FALSE), dans le cas contraire. Pour plus d'informations, reportez-vous à la section «IsMeasured» à la page 174.

Intérieur, extérieur et contour

Toutes les géométries occupent une position dans l'espace, définie par ce qu'il y a à l'intérieur, à l'extérieur et par leur contour. L'extérieur d'une géométrie est constitué de tout l'espace qu'elle n'occupe pas. Le contour sert d'interface entre l'intérieur et l'extérieur de la géométrie. L'intérieur consiste en l'espace occupé par la géométrie. Les sous-classes héritent directement des propriétés interieur et extérieur, mais chacune d'elle a un contour différent.

La fonction ST_Boundary utilise une géométrie en entrée et renvoie une géométrie représentant le contour de la géométrie source. Pour plus d'informations, reportez-vous à la section «ST_Boundary» à la page 197.

Simple ou complexe

Certaines sous-classes de géométries (lignes, multipoints et multilignes) peuvent être simples ou complexes. Une sous-classe est dite simple lorsqu'elle respecte toutes les règles topologiques imposées à la sous-classe, et complexe, dans la cas contraire. Une ligne est simple si elle ne forme pas d'intersection avec son intérieur. Un multipoint est simple si aucun des éléments le composant n'occupe le même espace de coordonnées. Une multiligne est simple si l'intérieur d'aucun de ses éléments ne forme une intersection avec son propre intérieur.

Le prédicat ST_IsSimple utilise une géométrie en entrée et renvoie une valeur 1 (TRUE) si la géométrie est simple et une valeur 0 (FALSE) dans le cas contraire. Pour plus d'informations, reportez-vous à la section «ST_IsSimple» à la page 244.

Vide ou non vide

Une géométrie est vide si elle ne contient aucun point. L'enveloppe, le contour, l'intérieur et l'extérieur d'une géométrie vide sont définis par la valeur NULL. Une géométrie vide est toujours simple et peut être dotée de

coordonnées Z ou de mesures. Les lignes et les multilignes vides ont une longueur 0. Les polygones et les multipolygones ont une surface 0.

Le prédicat ST_IsEmpty utilise une géométrie en entrée et renvoie une valeur 1 (TRUE) si la géométrie est vide et une valeur 0 (FALSE) dans le cas contraire. Pour plus d'informations, reportez-vous à la section «ST_IsEmpty» à la page 240.

Enveloppe

L'enveloppe d'une géométrie est la géométrique englobante formée par les coordonnées (X,Y) minimales et maximales. Hormis les exceptions ci-dessous, les enveloppes de la plupart des géométries forment un rectangle contour.

- L'enveloppe d'un point est le point lui-même car les coordonnées minimales et maximales sont identiques.
- L'enveloppe d'une ligne verticale ou horizontale est une ligne représentée par le contour (les extrémités) de la ligne source.

La fonction ST_Envelope utilise une géométrie en entrée et renvoie une géométrie englobante représentant son enveloppe. Pour plus d'informations, reportez-vous à la section «ST_Envelope» à la page 217.

Dimension

Une géométrie peut avoir une dimension de valeur 0, 1 ou 2. Ces dimensions sont répertoriées comme suit :

0 Sans longueur ni surface

1 Avec une longueur

2 Contenant une surface

Les sous-classes de points et multipoints ont une dimension 0. Les points représentent des entités dimensionnelles qui peuvent être modélisée par une seule coordonnées, alors que les sous-classes de multipoints représentent des données qui doivent être modélisées par un groupe de coordonnées indépendantes.

Les sous-classes de lignes et de multilignes ont une dimension 1. Elles stockent des segments de routes, des systèmes d'affluents et d'autres entités linéaires par nature.

Les sous-classes de polygones et de multipolygones ont une dimension 2. Les entités dont le périmètre entoure une surface définissable telle que les forêts, les parcelles de terrain et les plans d'eau peuvent être représentées par le type de données polygone ou multipolygone.

La dimension est importante non seulement en tant que propriété de la sous-classe, mais aussi parce qu'elle joue un rôle dans la détermination de la relation spatiale entre deux entités. La dimension de la ou des entité(s) résultante(s) conditionne le succès ou l'échec de l'opération. DB2 Extension Spatiale examine la dimension des entités pour déterminer le mode de comparaison à utiliser.

La fonction ST_Dimension utilise une géométrie en entrée et renvoie sa dimension sous forme d'entier. Pour plus d'informations, reportez-vous à la section «ST_Dimension» à la page 211.

Identificateur de système de références spatiales

Le système de références spatiales identifie la transformation à effectuer pour chaque géométrie.

Tous les systèmes de références spatiales connus de la base de données sont accessibles à partir de la vue DB2GSE.SPATIAL_REF_SYS du catalogue DB2 Extension Spatiale. Pour plus d'informations sur cette vue, reportez-vous au «Chapitre 11. Vues du catalogue» à la page 115.

La fonction ST_SRID utilise une géométrie en entrée et renvoie l'identificateur du système de références spatiales associé sous forme d'un entier. Pour plus d'informations, reportez-vous à la section «ST_SRID» à la page 280.

La fonction ST_Transform affecte une géométrie à un système de références spatiales différent de celui auquel elle est actuellement affectée. Pour plus d'informations, reportez-vous à la section «ST_Transform» à la page 285.

Géométries instanciables et fonctions associées

La présente section décrit les six sous-classes de géométries instanciables, ainsi que les fonctions associées. Il s'agit des sous-classes suivantes :

- Points
- Lignes
- Polygones
- Multipoints
- Multilignes
- Multipolygones

Pour l'illustration de la hiérarchie à laquelle appartiennent ces sous-classes et les symboles visuels qui leur sont associés, reportez-vous à la figure 9 à la page 130.

Points

Un point est une géométrie de dimension 0 qui occupe un seul emplacement dans l'espace des coordonnées. Un point comprend une abscisse (coordonnée X) et une ordonnée (Y) qui définissent son emplacement. Il peut également comporter une coordonnée Z et une mesure.

Un point est simple et a un contour de valeur NULL. Les points servent souvent à définir des entités telles que des puits de pétrole, des repères géographiques ou des altitudes.

Les fonctions uniquement exécutables sur la sous-classe des points sont les suivantes :

ST_Point
Utilise en entrée une abscisse (X), l'ordonnée (Y) associée et l'ID du système de références spatiales auquel appartiennent ces coordonnées, et renvoie le point défini par les coordonnées. Pour plus d'informations, reportez-vous à la section «ST_Point» à la page 271.

ST_CoordDim
Renvoie une valeur qui désigne les coordonnées contenues par un point et indique s'il contient également une mesure. Cette valeur est appelée *dimension de coordonnées*. Les dimensions de coordonnées possibles sont les suivantes :

2 Le point consiste en une abscisse et une ordonnée.

3 Le point consiste en une abscisse, une ordonnée et une coordonnée Z.

4 Le point consiste en une abscisse, une ordonnée, une coordonnée Z et une mesure.

Pour plus d'informations, reportez-vous à la section «ST_CoordDim» à la page 206.

ST_PointFromText
Utilise en entrée la représentation de type texte OGC WKB (well-known binary) d'un point et renvoie le point. Pour plus d'informations, reportez-vous à la section «ST_PointFromText» à la page 268.

ST_X Renvoie une valeur d'abscisse (X) de type de données ST_Point sous la forme d'un nombre à double précision. Pour plus d'informations, reportez-vous à la section «ST_X» à la page 292.

ST_Y Renvoie une valeur d'ordonnée (Y) de type de données ST_Point sous la forme d'un nombre à double précision. Pour plus d'informations, reportez-vous à la section «ST_Y» à la page 293.

Z Renvoie une valeur de coordonnée Z de type de données ST_Point sous la forme d'un nombre à double précision. Pour plus d'informations, reportez-vous à la section «Z» à la page 294.

M Renvoie une valeur de mesure de type de données ST_Point sous la forme d'un nombre à double précision. Pour plus d'informations, reportez-vous à la section «M» à la page 181.

Lignes

Une ligne est un objet unidimensionnel stocké comme une suite de points définissant un chemin linéaire interpolé. Une ligne est simple si elle ne se coupe pas elle-même. Les extrémités (contour) d'une ligne fermée occupent le même point dans l'espace. Une ligne est un anneau si elle est fermée et qu'elle ne se coupe pas. Outre les autres propriétés héritées de la géométrie de superclasse, les lignes sont dotées d'une longueur. Elles sont souvent utilisées pour définir des entités linéaires telles que des routes, des rivières ou des lignes électriques.

On appelle *anneau* une ligne simple dont le point initial et le point final sont identiques.

Les extrémités forment normalement le contour d'une ligne à moins que celle-ci ne soit fermée auquel cas le contour a la valeur NULL. L'intérieur d'une ligne est le chemin connecté situé entre les deux extrémités, sauf s'il est fermé auquel cas l'intérieur est continu.

Les fonctions exécutables sur les lignes sont les suivantes :

ST_StartPoint
Utilise une ligne en entrée et renvoie le premier point la constituant. Pour plus d'informations, reportez-vous à la section «ST_StartPoint» à la page 281.

ST_EndPoint
Utilise une ligne en entrée et renvoie le dernier point la constituant. Pour plus d'informations, reportez-vous à la section «ST_Endpoint» à la page 216.

ST_PointN
Utilise en entrée une ligne et un index jusqu'au *n*ème point, et renvoie ce point. Pour plus d'informations, reportez-vous à la section «ST_PointN» à la page 272.

ST_Length
Utilise une ligne en entrée et renvoie sa longueur sous la forme d'un nombre à double précision. Pour plus d'informations, reportez-vous à la section «ST_Length» à la page 247.

ST_NumPoints

Utilise une ligne en entrée et renvoie le nombre de points contenus dans la suite sous la forme d'un nombre entier. Pour plus d'informations, reportez-vous à la section «ST_NumPoints» à la page 262.

ST_IsRing

Utilise une ligne en entrée et renvoie une valeur 1 (TRUE) s'il s'agit d'un anneau et une valeur 0 (FALSE) dans le cas contraire. Pour plus d'informations, reportez-vous à la section «ST_IsRing» à la page 242.

ST_IsClosed

Utilise une ligne en entrée et renvoie une valeur 1 (TRUE) si la ligne est fermée et une valeur 0 (FALSE) dans le cas contraire. Pour plus d'informations, reportez-vous à la section «ST_IsClosed» à la page 238.

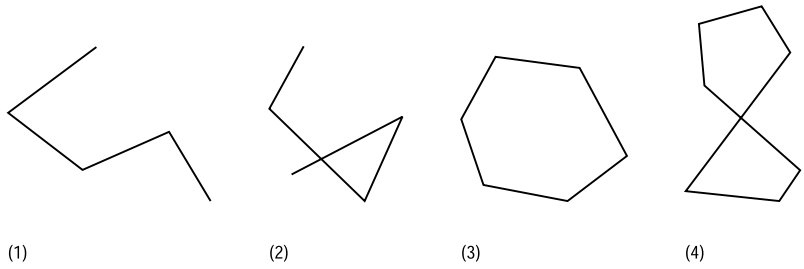


Figure 10. Objets de type ligne.

1. Ligne simple non fermée
2. Ligne complexe non fermée
3. Ligne simple fermée formant par conséquent un anneau
4. Ligne complexe fermée. Il ne s'agit pas d'un anneau.

Polygones

Un polygone est une surface bidimensionnelles stockée sous la forme d'une séquence de points définissant son anneau de contour externe et 0 ou plus anneaux internes. Les anneaux d'un polygone ne peuvent pas se chevaucher. Par conséquent et par définition, les polygones sont toujours simples. La plupart du temps, ils définissent des parcelles de terrain, des plans d'eau et autres entités dotées d'un domaine spatial.

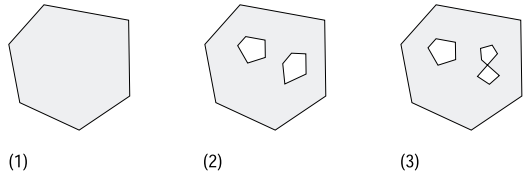


Figure 11. Polygones.

1. Polygone dont le contour est défini par un anneau externe.
2. Polygone dont le contour est défini par un anneau externe et deux anneaux internes. La zone située à l'intérieur des anneaux internes fait partie de l'extérieur des polygones.
3. Polygone légal car les anneaux forment une intersection à un seul point tangent.

L'anneau extérieur et le ou les anneau(x) intérieur(s) définissent le contour du polygone, et l'espace situé entre les anneaux, l'intérieur. Les anneaux d'un polygone peuvent former une intersection à un point tangent mais ils ne peuvent jamais se chevaucher. Outre les autres propriétés héritées de la géométrie de superclasse, les polygones sont dotées d'une surface.

Les fonctions exécutables sur les polygones sont les suivantes :

ST_Area
Utilise un polygone en entrée et renvoie sa surface sous la forme d'un nombre à double précision. Pour plus d'informations, reportez-vous à la section «ST_Area» à la page 193.

ST_ExteriorRing
Utilise un polygone en entrée et renvoie son anneau extérieur sous la forme d'une ligne. Pour plus d'informations, reportez-vous à la section «ST_ExteriorRing» à la page 220.

ST_NumInteriorRing
Utilise un polygone en entrée et renvoie le nombre d'anneaux intérieurs qu'il contient. Pour plus d'informations, reportez-vous à la section «ST_NumInteriorRing» à la page 261.

ST_InteriorRingN
Utilise un polygone et un index en entrée, et renvoie le *n*ème anneau intérieur sous la forme d'une ligne. Pour plus d'informations, reportez-vous à la section «ST_InteriorRingN» à la page 229.

ST_Centroid
Utilise un polygone en entrée et renvoie un point constituant le centre de la superficie du polygone. Pour plus d'informations, reportez-vous à la section «ST_Centroid» à la page 201.

ST_PointOnSurface
Utilise un polygone en entrée et renvoie un point qui se trouve avec certitude sur la surface du polygone. Pour plus d'informations, reportez-vous à la section «ST_PointOnSurface» à la page 273.

ST_Perimeter
Utilise un polygone en entrée et renvoie le périmètre de sa surface. Pour plus d'informations, reportez-vous à la section «ST_Perimeter» à la page 267.

Multipoints

Un multipoint est une collection de points et, à l'instar des éléments qui le composent, sa dimension est de 0. Un multipoint est simple si aucun de ses éléments n'occupe le même espace de coordonnées. Le contour d'un multipoint a une valeur NULL. Les multipoints peuvent être utilisés pour définir des phénomènes tels que les cas déclarés au début d'une épidémie, etc.

Les fonctions exécutables sur les multipoints sont les suivantes :

ST_NumGeometries
Utilise une collection homogène en entrée et renvoie le nombre d'éléments géométriques de base qu'elle contient. Pour plus d'informations, reportez-vous à la section «ST_NumGeometries» à la page 260.

ST_GeometryN
Utilise une collection homogène et un index en entrée, et renvoie la nème géométrie de base. Pour plus d'informations, reportez-vous à la section «ST_GeometryN» à la page 226.

Multilignes

Une multiligne est une collection de lignes. Les multilignes sont simples si elles ne forment d'intersections qu'aux extrémités des lignes la composant. Elle sont complexes s'il existe des points d'intersection à l'intérieur des lignes.

Le contour d'une multiligne est constitué par les extrémités non intersectées des lignes. La multiligne est fermée si toutes les lignes qui la composent sont fermées. Le contour d'une multiligne a une valeur NULL si toutes les extrémités des éléments de base génèrent des intersections. Outre les autres propriétés héritées de la géométrie de superclasse, les multilignes sont dotées d'une longueur. Elle permettent de définir des cours d'eau ou des réseaux routiers.

Les fonctions exécutables sur les multilignes sont les suivantes :

ST_Length
Utilise une multiligne en entrée et renvoie la longueur cumulée de toutes les lignes sous la forme d'un nombre à double précision. Pour plus d'informations, reportez-vous à la section «ST_Length» à la page 247.

ST_IsClosed
Utilise une multiligne en entrée et renvoie une valeur 1 (TRUE) si la

multiligne est fermée et une valeur 0 (FALSE) dans le cas contraire. Pour plus d'informations, reportez-vous à la section «ST_IsClosed» à la page 238.

ST_NumGeometries
Utilise une collection homogène en entrée et renvoie le nombre d'éléments géométriques de base qu'elle contient. Pour plus d'informations, reportez-vous à la section «ST_NumGeometries» à la page 260.

ST_GeometryN
Utilise une collection homogène et un index en entrée, et renvoie la nème géométrie de base. Pour plus d'informations, reportez-vous à la section «ST_GeometryN» à la page 226.

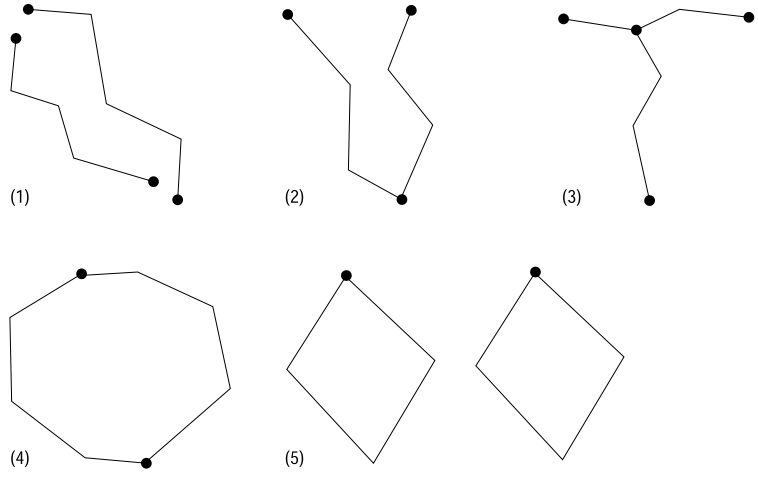


Figure 12. Multilignes.

1. Multiligne simple dont le contour est défini par les quatre extrémités des deux lignes qui la composent.
2. Multiligne simple car seules les extrémités des lignes génèrent des intersections. Le contour est défini par les deux extrémités qui ne forment pas d'intersection.
3. Multiligne complexe car il y a un point d'intersection à l'intérieur d'une des lignes. Le contour de cette multiligne est défini par quatre extrémités, dont le point d'intersection.
4. Multiligne simple non fermée. Elle n'est pas fermée car les lignes qui la composent ne le sont pas et elle est simple parce qu'il n'existe aucun point d'intersection à l'intérieur des lignes.
5. Multiligne simple fermée. Elle est fermée car tous les éléments qui la composent le sont et elle est simple parce qu'il n'existe aucune intersection à l'intérieur des lignes.

Multipolygones

Le contour d'un multipolygone consiste en la longueur cumulée des anneaux extérieur et intérieur des éléments qui la composent. L'intérieur d'un

multipolygone est défini comme le cumul des intérieurs des polygones le composant. Le contour des éléments d'un multipolygone ne peut former une intersection qu'à un point tangent. Outre les autres propriétés héritées de la géométrie de superclasse, les multipolygones sont dotées d'une surface. Ils définissent des entités telles que les strates d'une forêt ou une parcelle de terrain non contiguë telle qu'un chapelet d'îles.

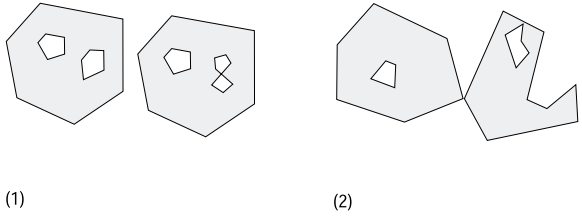


Figure 13. Multipolygones.

1. Multipolygone à deux polygones. Le contour est défini par les deux anneaux externes et les trois anneaux internes.
2. Multipolygone à deux polygones. Le contour est défini par les deux anneaux externes et les deux anneaux internes. Les deux polygones forment une intersection à un point tangent.

Les fonctions exécutables sur les multipolygones sont les suivantes :

ST_Area
Utilise un multipolygone en entrée et renvoie la surface cumulée de tous les polygones sous la forme d'un nombre à double précision. Pour plus d'informations, reportez-vous à la section «ST_Area» à la page 193.

ST_Centroid
Utilise un multipolygone en entrée et renvoie un point représentant le centre géométrique pondéré. Pour plus d'informations, reportez-vous à la section «ST_Centroid» à la page 201.

ST_NumGeometries
Utilise une collection homogène en entrée et renvoie le nombre d'éléments géométriques de base qu'elle contient. Pour plus d'informations, reportez-vous à la section «ST_NumGeometries» à la page 260.

ST_GeometryN
Utilise une collection homogène et un index en entrée, et renvoie la nème géométrie de base. Pour plus d'informations, reportez-vous à la section «ST_GeometryN» à la page 226.

Fonctions permettant de visualiser des relations et des comparaisons, de générer des géométries et de convertir des formats des valeurs

Les sections précédentes ont présenté les catégories de fonctions spatiales suivantes :

- Fonctions associées aux propriétés des géométries
- Fonctions associées à des géométries spécifiques

La présente section décrit les trois catégories supplémentaires suivantes :

- Fonctions déterminant des modes de relation et de comparaison entre entités géographiques
- Fonctions de génération de nouvelles géométries
- Fonctions de conversion des valeurs d'une géométrie dans un format importable ou exportable

Fonctions de visualisation de relations et de comparaison entre entités géographiques

Plusieurs fonctions spatiales renvoient des informations sur les relations existant entre entités géographiques ou sur leur comparaison. Il s'agit dans la plupart des cas de fonctions booléennes et elles sont appelées *prédicats*. La présente section fait un tour d'horizon des prédicats, puis les traite séparément.

Prédicats

Les prédicats renvoient une valeur 1 (TRUE) si une comparaison satisfait aux critères de la fonction ou 0 (FALSE) dans le cas contraire. Les prédicats qui testent une relation spatiale comparent deux par deux des géométries qui peuvent s'avérer de dimension ou de type différent.

Les prédicats comparent les abscisses (X) et les ordonnées (Y) des géométries soumises au test. Les coordonnées Z et la mesure sont ignorées (si elles existent). Cela permet de comparer des géométries dotées de coordonnées Z ou de mesure à des géométries qui en sont dépourvues.

Le modèle *DE-9IM (Dimensionally Extended 9 Intersection Model)* [1] est une approche mathématique qui définit les relations spatiales existant entre géométries de types et de dimensions différents, prises deux par deux. Ce

1. Le modèle DE-91M a été développé par Clementini et Felice, qui ont agrandi les dimensions du modèle à 9 intersections de Egenhofer et Herring. Il est le fruit de la collaboration de quatre auteurs : Clementini, Eliseo, Di Felice et van Osstrom. Ils l'ont publié dans le document intitulé "A Small Set of Formal Topological Relationships Suitable for End-User Interaction," D. Abel and B.C. Ooi (Ed.), *Advances in Spatial Database—Third International Symposium. SSD '93*. LNCS 692. Pp. 277-295. Le modèle à 9 intersections de M.J. Egenhofer et J. Herring (Editions Springer-Verlag Singapore 1993) a été publié dans le document intitulé "Categorizing binary topological relationships between regions, lines, and points in geographic databases," *Tech. Report, Department of Surveying Engineering*, University of Maine, Orono, ME 1991.

modèle exprime les relations spatiales entre tous les types de géométries sous la forme des intersections de l'intérieur, du contour et de l'extérieur de ces géométries prises deux par deux, en tenant compte de la dimension des intersections obtenues.

Considérons les géométries *a* et *b* : I(*a*), B(*a*) et E(*a*) représentent respectivement l'intérieur, le contour et l'extérieur de *a*, et I(b), C(b) et E(b) ceux de *b*. Les intersections entre I(a), C(a) et E(a), et I(b), C(b) et E(b) génèrent une matrice 3 par 3. Chaque intersection peut donner des géométries de dimensions différentes. Par exemple, l'intersection des contours de deux polygones consiste en un point et une ligne, auquel cas la fonction DIM renverrait la dimension maximale de 1.

La fonction dim renvoie une valeur 1, 0, 1 ou 2. 1 correspond à l'ensemble de valeurs NULL ou à dim(null), renvoyé lorsqu'aucune intersection n'a été détectée.

	Intérieur	**Contour**	**Extérieur**
Intérieur	dim(I(a) ∩ I(b))	dim(I(a) ∩ C(b))	dim(I(a) ∩ E(b))
Contour	dim(C(a) ∩ I(b))	dim(C(a) ∩ C(b))	dim(C(a) ∩ E(b))
Extérieur	dim(E(a) ∩ I(b))	dim(E(a) ∩ C(b))	dim(E(a) ∩ E(b))

Le résultat des prédicats de relation spatiale peut être interprété ou vérifié en le comparant à une matrice de schémas, qui représente les valeurs acceptables pour le modèle DE-9IM.

La matrice de schémas contient les valeurs acceptables pour chacune des cellules d'intersection de la matrice. Les schémas possibles sont les suivants :

T Il doit y avoir une intersection, dim = 0, 1 ou 2.

F Il ne doit pas y avoir d'intersection, dim = -1.

***** La présence ou l'absence d'intersection importe peu, dim = -1, 0, 1 ou 2.

0 Il doit y avoir une intersection et sa dimension maximale doit être de 0, dim = 0.

1 Il doit y avoir une intersection et sa dimension maximale doit être de 1, dim = 1.

2 Il doit y avoir une intersection et sa dimension maximale doit être de 2, dim = 2.

Par exemple, la matrice de schémas ci-après associée au prédicat ST_Within comprend les valeur T, F et *.

Tableau 40. Matrice du prédicat ST_Within. Matrice de schémas du prédicat ST_Within pour les combinaisons de géométries

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	T	*	F
	Contour	*	*	F
	Extérieur	*	*	*

Le prédicat ST_Within renvoie la valeur TRUE lorsque les intérieurs des deux géométries forment une intersection, et que l'intérieur et le contour de *a* ne coupe pas l'extérieur de *b.* Toutes les autres conditions n'ont pas d'importance.

Chaque prédicat est associé à une matrice de schémas au moins, mais certaines en exigent plusieurs pour décrire les relations des différentes combinaisons de types de géométries.

ST_Equals

ST_Equals renvoie la valeur 1 (TRUE) si deux géométries de même type ont des valeurs d'abscisse (X) et d'ordonnée (Y) identiques.

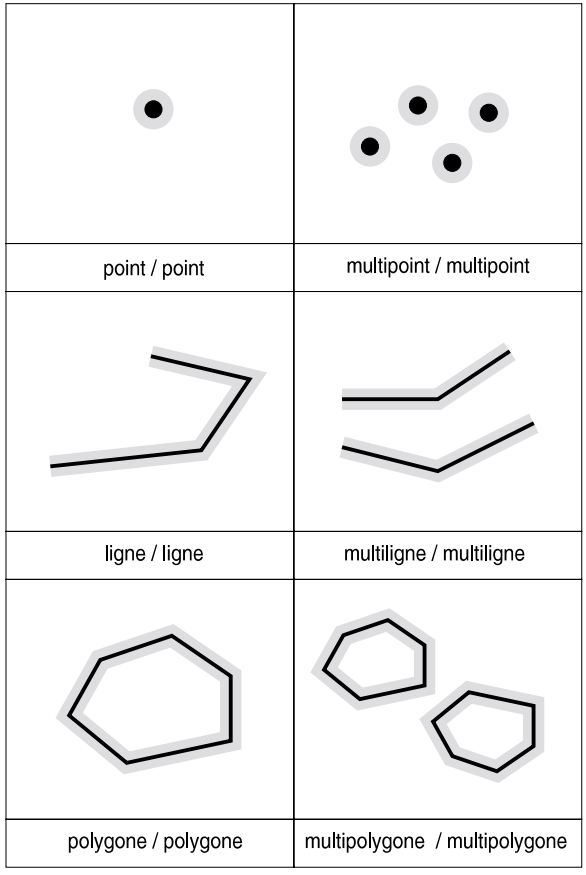


Figure 14. ST_Equals. Des géométries sont égales si leurs abscisses et leurs ordonnées sont identiques.

Tableau 41. Matrice d'égalité. La matrice de schémas DE-9IM associée au prédicat d'égalité vérifie que les intérieurs génèrent une intersection mais qu'aucune partie de l'intérieur ou du contour de l'une ou l'autre des géométries ne coupe l'extérieur de l'autre.

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	T	*	F
	Contour	*	*	F
	Extérieur	F	F	*

Pour plus d'informations, reportez-vous à la section «ST_Equals» à la page 219.

ST_OrderingEquals

ST_OrderingEquals compare deux géométries et renvoie la valeur 1 (TRUE) si elles sont égales et que les coordonnées sont dans le même ordre ; sinon, ce prédicat renvoie la valeur 0 (FALSE). Pour plus d'informations, reportez-vous à la section «ST_OrderingEquals» à la page 263.

ST_Disjoint

ST_Disjoint renvoie la valeur 1 (TRUE) si l'intersection entre deux géométries est un ensemble vide.

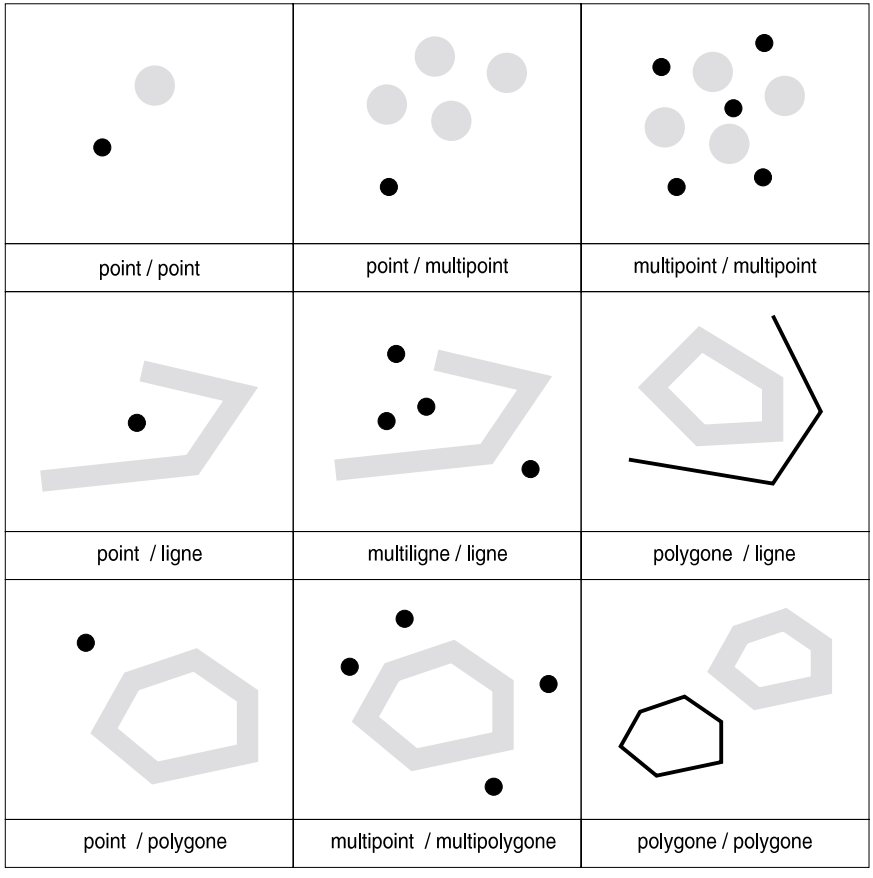


Figure 15. ST_Disjoint. Des géométries sont disjointes si elles ne forment aucune intersection.

Tableau 42. Matrice du prédicat ST_Disjoint. La matrice de schémas du prédicat ST_Disjoint indique simplement que les intérieurs ou les contours des deux géométries ne forment aucune intersection.

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	F	F	*
	Contour	F	F	*
	Extérieur	*	*	*

Pour plus d'informations, reportez-vous à la section «ST_Disjoint» à la page 213.

ST_Intersects

ST_Intersects renvoie la valeur 1 (TRUE) si l'intersection n'est pas un ensemble vide. Il renvoie le résultat exactement inverse par rapport au prédicat ST_Disjoint.

Le prédicat ST_Intersects renvoie la valeur TRUE si les conditions de l'une des matrices de schémas suivantes renvoie cette même valeur.

Tableau 43. Matrice du prédicat ST_Intersects (1). Le prédicat ST_Intersects renvoie la valeur TRUE si les interieurs des deux géométries se coupent.

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	T	*	*
	Contour	*	*	*
	Extérieur	*	*	*

Tableau 44. Matrice associée au prédicat ST_Intersects (2). Le prédicat ST_Intersects renvoie la valeur TRUE si le contour de la première géométrie coupe celui de la seconde.

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	*	T	*
	Contour	*	*	*
	Extérieur	*	*	*

Tableau 45. Matrice associée au prédicat ST_Intersects (3). Le prédicat ST_Intersects renvoie la valeur TRUE si le contour de la première géométrie coupe l'intérieur de la seconde.

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	*	*	*
	Contour	T	*	*
	Extérieur	*	*	*

Tableau 46. Matrice associée au prédicat ST_Intersects (4). Le prédicat ST_Intersects renvoie la valeur TRUE si les contours des deux géométries se coupent.

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	*	*	*
	Contour	*	T	*
	Extérieur	*	*	*

Pour plus d'informations, reportez-vous à la section «ST_Intersects» à la page 237.

EnvelopesIntersect

Cette fonction renvoie la valeur 1 (TRUE) si les enveloppes de deux géométries forment une intersection. Il s'agit d'une fonction pratique qui implémente efficacement le prédicat ST_Intersects (`ST_Envelope(g1),ST_Envelope(g2)`). Pour plus d'informations, reportez-vous à la section «EnvelopesIntersect» à la page 171.

ST_Touches

ST_Touches renvoie la valeur 1 (TRUE) si aucun des points communs aux deux géométries ne forme une intersection avec les intérieurs des deux géométries. Au moins l'une des deux géométries doit être une ligne, un polygone, une multiligne ou un multipolygone.

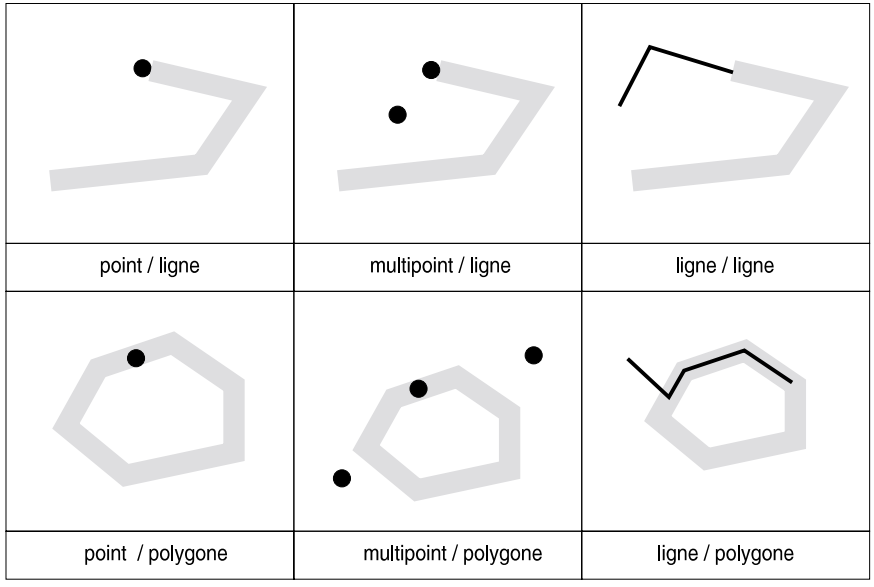


Figure 16. ST_Touches

Les matrices de schémas indiquent que le prédicat ST_Touches renvoie la valeur TRUE lorsque les intérieurs de la géométrie ne se coupent pas et que la contour de l'une des deux figures génère une intersection avec l'intérieur ou le contour de l'autre.

Tableau 47. Matrice du prédicat ST_Touches (1)

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	F	T	*
	Contour	*	*	*
	Extérieur	*	*	*

Tableau 48. Matrice du prédicat ST_Touches (2)

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	F	*	*
	Contour	T	*	*
	Extérieur	*	*	*

Tableau 49. Matrice du prédicat ST_Touches (3)

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	F	*	*
	Contour	*	T	*
	Extérieur	*	*	*

Pour plus d'informations, reportez-vous à la section «ST_Touches» à la page 284.

ST_Overlaps

ST_Overlaps compare deux géométries de même dimension. Il renvoie la valeur 1 (TRUE) si l'ensemble de leur intersection donne une géométrie différente des deux mais de dimension identique.

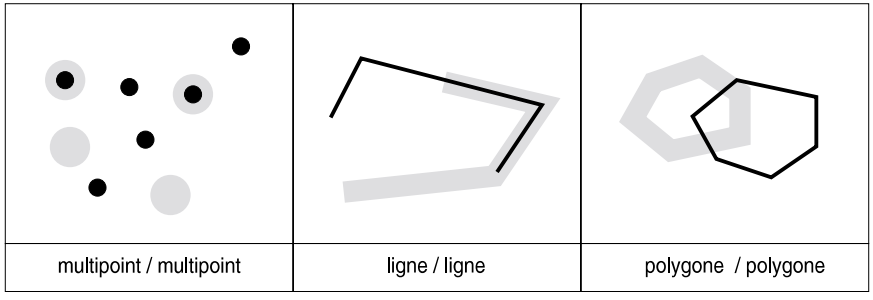


Figure 17. ST_Overlaps

La matrice de schémas illustrée au tableau 50 s'applique aux chevauchements de type polygone/polygone, multipoint/multipoint et multipolygone/multipolygone. Pour ces combinaisons, le prédicat de chevauchement renvoie la valeur TRUE si l'intérieur des deux géométries forme une intersection avec l'intérieur et l'extérieur de l'autre.

Tableau 50. Matrice du prédicat ST_Overlaps (1)

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	T	*	T
	Contour	*	*	*
	Extérieur	T	*	*

La matrice de schémas illustrée au tableau 51 à la page 152 s'applique aux chevauchements de type multiligne/multiligne. Dans ce cas, l'intersection des géométries doit avoir pour résultat une géométrie de dimension 1 (une autre ligne). Si la dimension de l'intersection des intérieurs est de 1, le prédicat ST_Overlaps renvoie la valeur FALSE alors que le prédicat ST_Crosses renvoie la valeur

TRUE.

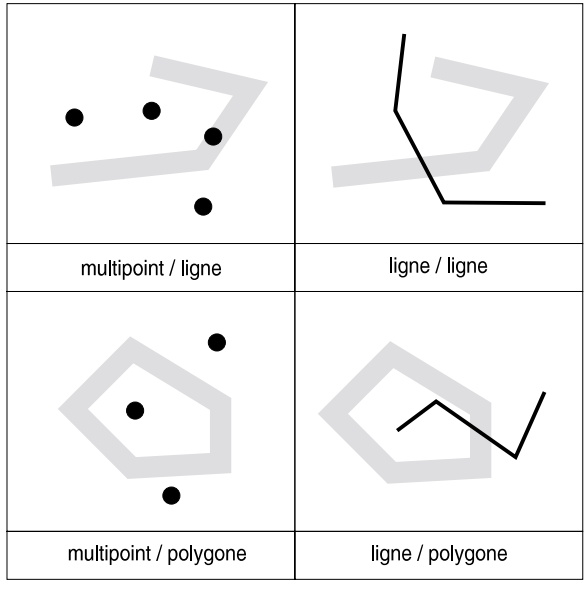
Tableau 51. Matrice du prédicat ST_Overlaps (2)

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	1	*	T
	Contour	*	*	*
	Extérieur	T	*	*

Pour plus d'informations, reportez-vous à la section «ST_Overlaps» à la page 265.

ST_Crosses

ST_Crosses renvoie la valeur 1 (TRUE) si l'intersection donne une géométrie dont la dimension est inférieure d'une unité par rapport à la dimension maximale des deux géométries source, et que l'ensemble de l'intersection est à l'intérieur de ces deux géométries. ST_Crosses ne renvoie la valeur 1 (TRUE) que pour les comparaisons de type multipoint/polygone, multipoint/ligne, ligne/ligne, ligne/polygone et ligne/multipolygone.



La matrice de schémas illustrée au tableau 52 à la page 153 s'applique aux comparaisons de types multipoint/ligne, multipoint/multiligne, multipoint/polygone, multipoint/multipolygone, ligne/polygone, ligne/multipolygone. Elle indique que les intérieurs doivent se couper et que l'intérieur de la première (géometrie *a*) doit former une intersection avec l'extérieur de la seconde (géométrie *b*).

Tableau 52. Matrice du prédicat ST_Crosses (1)

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	T	*	T
	Contour	*	*	*
	Extérieur	*	*	*

La matrice illustrée au tableau 53 s'applique aux comparaisons de type ligne/ligne, ligne/multiligne et multiligne/multiligne. Elle indique que la dimension de l'intersection des intérieurs doit être de 0 (intersection en un point). Si cette dimension est de 1 (intersection en une ligne), le prédicat ST_Crosses renvoie la valeur FALSE alors que le prédicat ST_Overlaps renvoie la valeur TRUE.

Tableau 53. Matrice du prédicat ST_Crosses (2)

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	0	*	*
	Contour	*	*	*
	Extérieur	*	*	*

Pour plus d'informations, reportez-vous à la section «ST_Crosses» à la page 208.

ST_Within

ST_Within renvoie la valeur 1 (TRUE) si la première géométrie est entièrement contenue dans la seconde. Ce prédicat renvoie le résultat exactement inverse par rapport au prédicat ST_Contains.

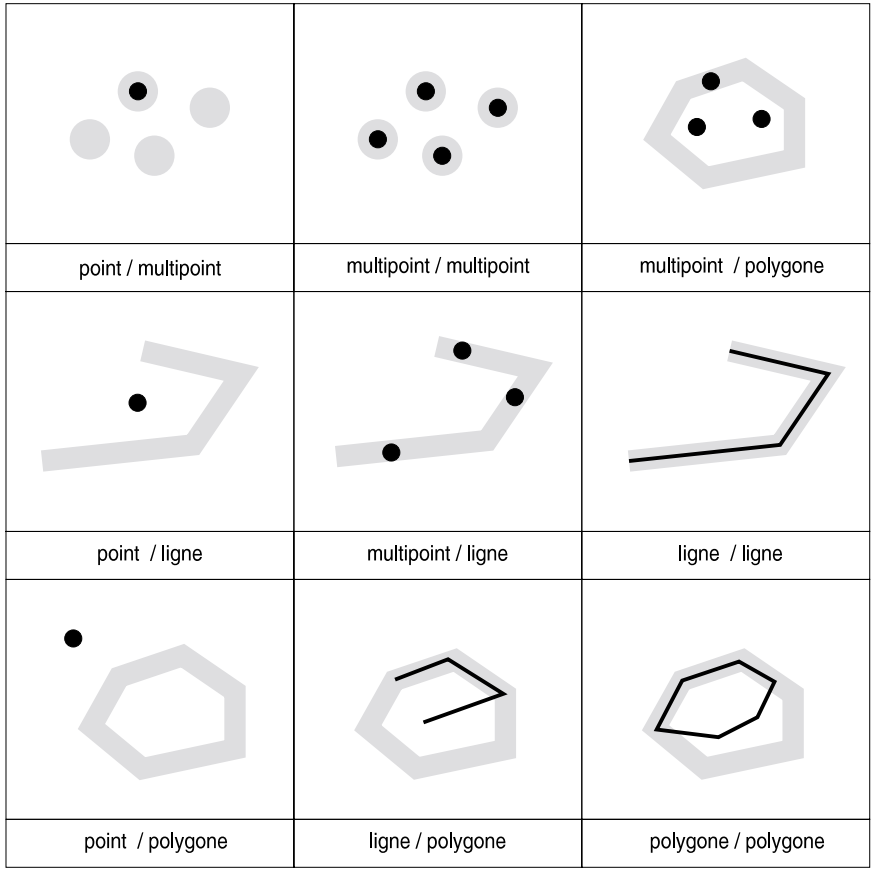


Figure 18. Within

La matrice de schémas associée au prédicat ST_Within indique que les intérieurs des deux géométries doivent former une intersection, et que l'intérieur et le contour de la première (géométrie *a*) ne doivent pas générer d'intersection avec l'extérieur de la seconde (géométrie *b*).

Tableau 54. Matrice du prédicat ST_Within

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	T	*	F
	Contour	*	*	F
	Extérieur	*	*	*

Pour plus d'informations, reportez-vous à la section «ST_Within» à la page 287.

ST_Contains

ST_Contains renvoie la valeur 1 (TRUE) si la seconde géométrie est entièrement contenue dans la première. Ce prédicat renvoie le résultat exactement inverse par rapport au prédicat ST_Within.

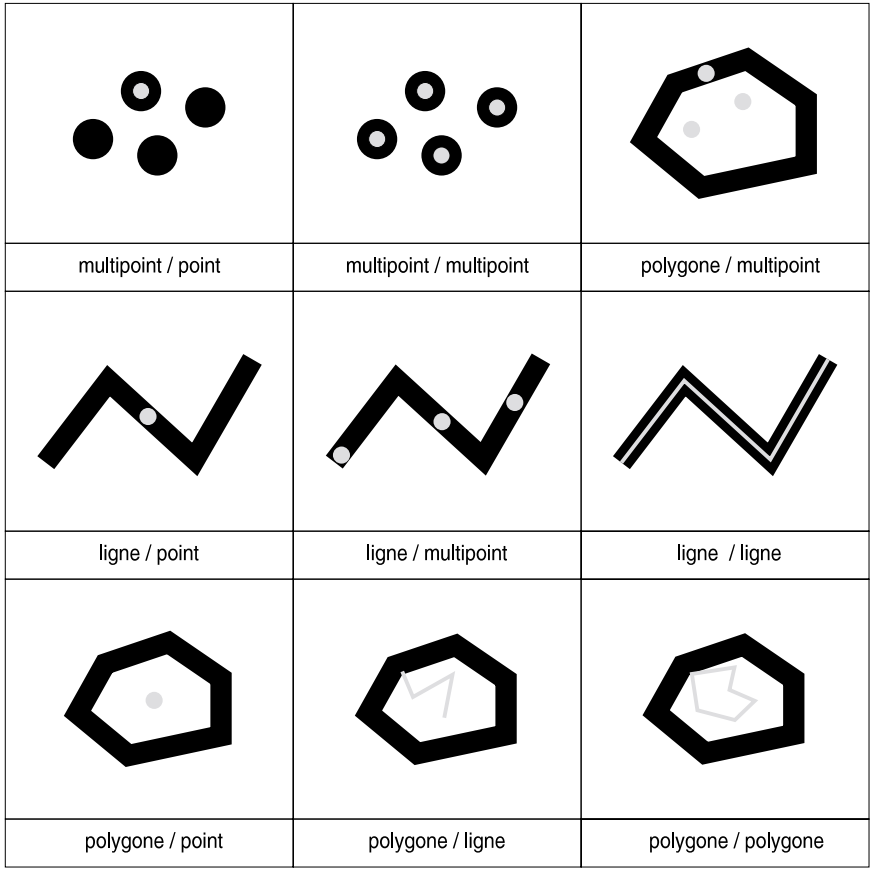


Figure 19. ST_Contains

La matrice de schémas associée au prédicat ST_Contains indique que les intérieurs des deux géométries doivent former une intersection, et que l'intérieur et le contour de la seconde (géométrie *b*) ne doivent pas générer d'intersection avec l'extérieur de la première (géométrie *a*).

Tableau 55. Matrice du prédicat ST_Contains

		b		
		Intérieur	**Contour**	**Extérieur**
a	**Intérieur**	T	*	*
	Contour	*	*	*
	Extérieur	F	F	*

Pour plus d'informations, reportez-vous à la section «ST_Contains» à la page 202.

ST_Relate

ST_Relate compare deux géométries et renvoie la valeur 1 (TRUE) si elles remplissent les conditions spécifiées par la chaîne de la matrice DE-91M ; sinon, ce prédicat renvoie la valeur 0 (FALSE). Pour plus d'informations, reportez-vous à la section «ST_Relate» à la page 278.

ST_Distance

La fonction ST_Distance indique la distance la plus courte séparant deux entités disjointes. Si les entités ne sont pas disjointes, la fonction renvoie une distance minimale de valeur 0.

Par exemple, ST_Distance peut indiquer la distance la plus courte qu'un avoin doit parcourir pour aller d'un point à un autre. La figure 3 à la page 7, représente cette information.

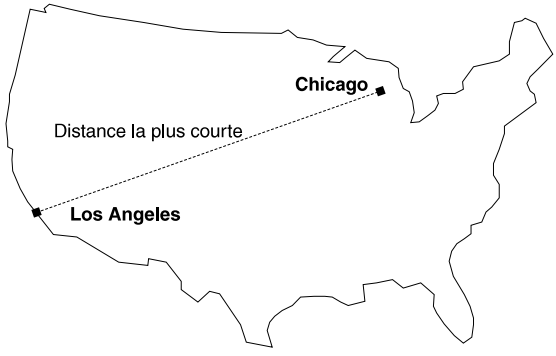


Figure 20. Distance minimale entre deux villes. ST_Distance peut utiliser les coordonnées correspondant aux positions de Los Angeles et de Chicago en tant qu'entrée, et renvoyer une valeur désignant la distance minimale entre ces deux positions.

Pour plus d'informations, reportez-vous à la section «ST_Distance» à la page 215.

Fonctions de génération de nouvelles géométries à partir de géométries existantes

DB2 Extension Spatiale fournit des prédicats et des fonctions de transformation qui génèrent de nouvelles géométries à partir de géométries existantes.

ST_Intersection

La fonction ST_Intersection renvoie l'ensemble d'intersection de deux géométries. Cet ensemble est toujours renvoyé sous la forme d'une collection dotée de la dimension minimale des géométries source. Par exemple, pour une ligne qui forme une intersection avec un polygone, la fonction ST_intersection renvoie une multiligne formée de la partie de la ligne commune avec l'intérieur et le contour du polygone. La multiligne contient plusieurs lignes si la ligne source forme une intersection avec le polygone en plusieurs segments discontinus. Si les géométries ne se coupent pas ou si l'intersection donne une dimension inférieure aux deux géométries source, la fonction renvoie une géométrie vide.

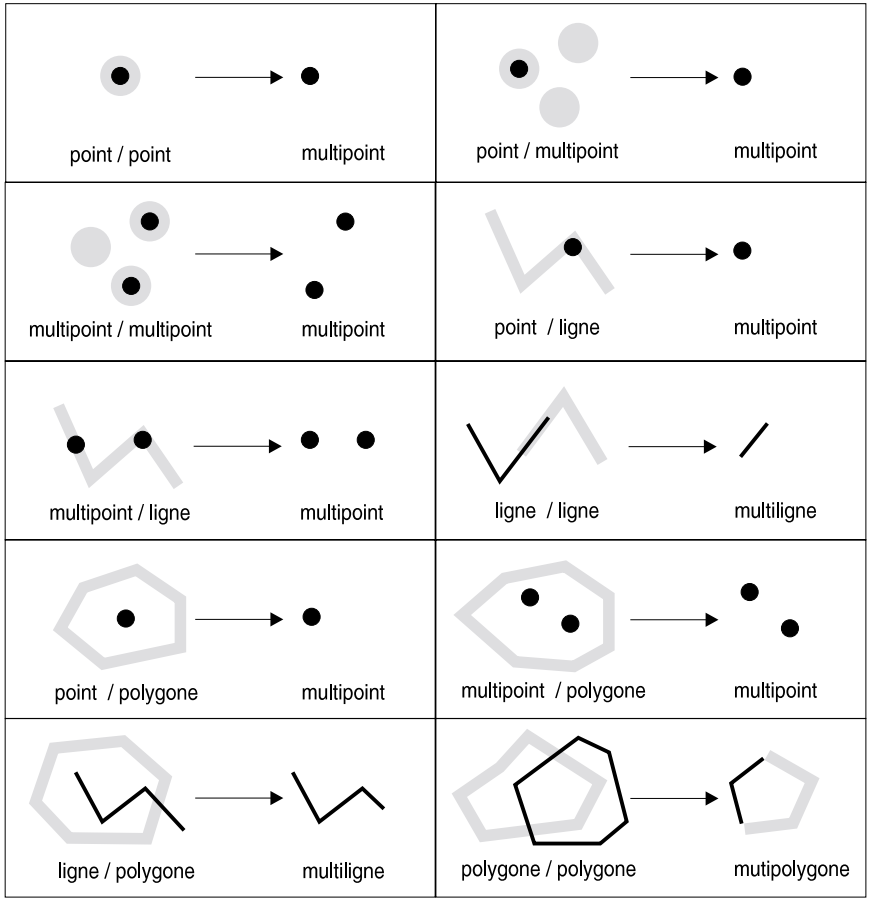


Figure 21. ST_Intersection. Exemples de fonction ST_Intersection

Pour plus d'informations, reportez-vous à la section «ST_Intersection» à la page 235.

ST_Difference

La fonction ST_Difference renvoie la partie de la première géométrie qui ne forme pas d'intersection avec la seconde géométrie. Il s'agit du ET NON logique de l'espace. La fonction ST_Difference ne fonctionne qu'avec des géométries de dimension identique et elle renvoie une collection de la même dimension que les géométries source. En cas d'égalité des géométries, la fonction renvoie une géométrie vide.

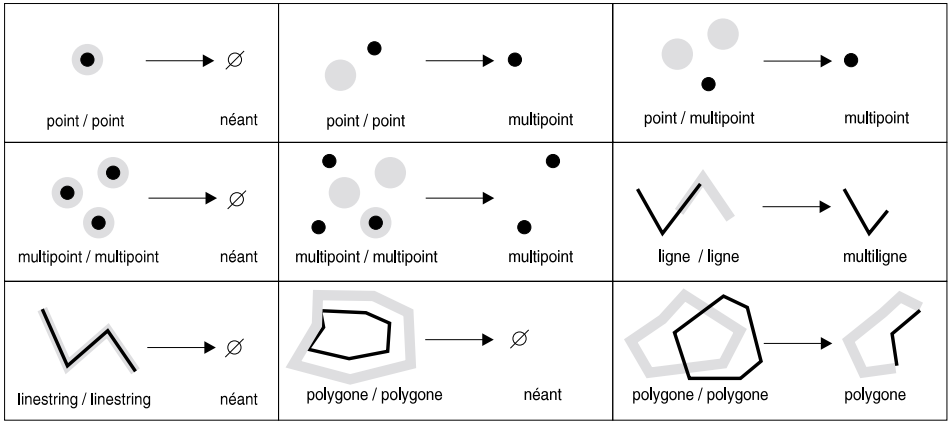


Figure 22. ST_Difference

Pour plus d'informations, reportez-vous à la section «ST_Difference» à la page 210 .

ST_Union

La fonction ST_Union renvoie l'ensemble d'union de deux géométries. Il s'agit du OU logique de l'espace. Les géométries source doivent être de même dimension. ST_Union renvoie toujours une collection en tant que résultat.

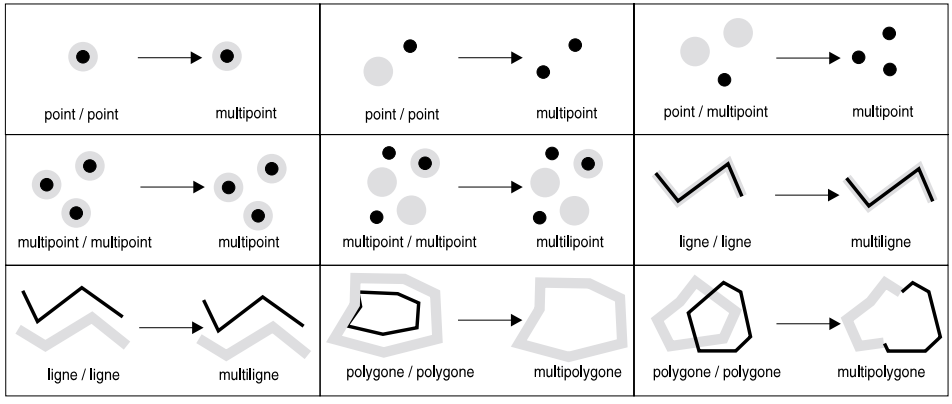


Figure 23. ST_Union

Pour plus d'informations, reportez-vous à la section «ST_Union» à la page 286.

ST_Buffer

La fonction ST_Buffer génère une géométrie en encerclant une géométrie à une distance déterminée. On obtient un polygone lorsqu'on exécute la fonction ST_Buffer sur une géométrie principale ou que les éléments d'une collection sont suffisamment rapprochés pour que tous les polygones tampons se chevauchent. Cependant, lorsque les éléments d'une collection, sur laquelle est exécutée la fonction ST_Buffer, sont suffisamment éloignés les uns des autres, on obtient des polygones tampons distincts, auquel cas la fonction ST_Buffer renvoie un multipolygone.

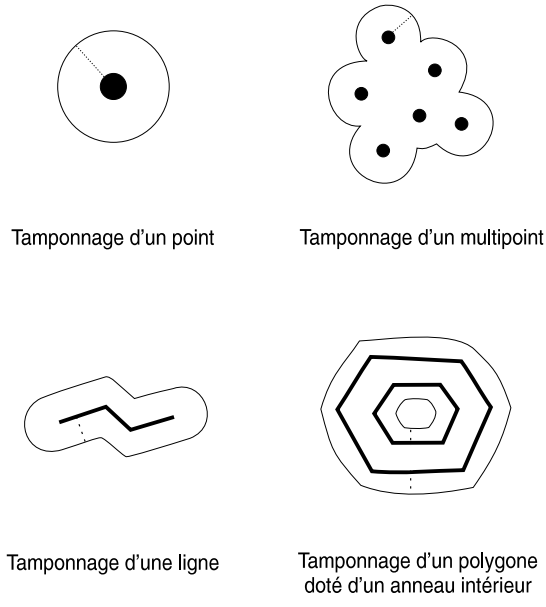


Figure 24. ST_Buffer

La fonction ST_Buffer accepte les distances positives et négatives. Toutefois, seules les géométries dotées d'une dimension de valeur 2 (polygones et multipolygones) appliquent un tampon négatif. La valeur absolue de la distance tampon est utilisée dès que la dimension de la géométrie source est inférieure à 2 (toutes les géométries autres que les polygones ou multipolygones).

En règle générale, pour les anneaux extérieurs, les distances tampons positives génèrent des anneaux de polygone qui sont éloignés du centre de la géométrie source, et les distances négatives, des anneaux de polygone ou de multipolygone plus proches du centre. Pour les anneaux intérieurs d'un polygone ou d'un multipolygone, une distance tampon positive génère un anneau tampon proche du centre et une distance négative, un anneau tampon éloigné du centre.

Le processus de tamponnage fusionne les polygones qui se chevauchent. Les distances négatives supérieures à la moitié de la largeur intérieure maximale d'un polygone génèrent une géométrie vide.

Pour plus d'informations, reportez-vous à la section «ST_Buffer» à la page 199.

LocateAlong

Pour les géométries dotées de mesures, la fonction LocateAlong permet de trouver l'emplacement d'une mesure déterminée. Elle renvoie celui-ci sous la

forme d'un multipoint. Si la dimension de la géométrie source est de 0 (par exemple, un point ou un multipoint), il doit s'agir d'une correspondance exacte et les points ayant une mesure identique sont renvoyés sous la forme d'un multipoint. Toutefois, pour les géométries source dont la dimension est supérieure à 0, l'emplacement est interpolé. Par exemple, si la mesure entrée est 5,5 et que les mesures des sommets d'une ligne sont respectivement de 3, 4, 5, 6 et 7, la fonction renvoie le point interpolé qui se situe exactement à mi-chemin entre les sommets dotés des mesures 5 et 6.

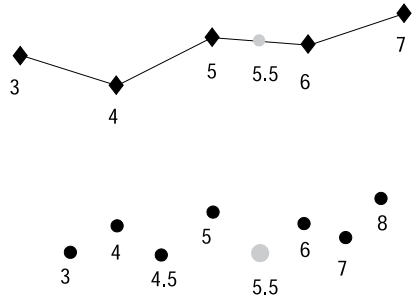


Figure 25. LocateAlong

Pour plus d'informations, reportez-vous à la section «LocateAlong» à la page 177.

LocateBetween

La fonction LocateBetween renvoie l'ensemble des chemins ou positions situés entre deux mesures provenant d'une géométrie source dotées de mesures. Si la dimension de la géométrie source est de 0, LocateBetween renvoie un multipoint contenant tous les points dont les mesures se trouvent entre les deux mesures source. Dans le cas de géométries source ayant une dimension supérieure à 0, LocateBetween renvoie une multiligne si un chemin peut être interpolé ; si tel n'est pas le cas, la fonction renvoie un multipoint contenant les positions des points. LocateBetween renvoie un point vide lorsqu'elle ne parvient pas à interpoler un chemin ou à détecter une position entre les mesures. LocateBetween exécute une recherche inclusive des géométries ; par conséquent, les mesures des géométries doivent être supérieures ou égales à la mesure *à partir de* (from) et inférieures ou égales à la mesure *jusqu'à* (to).

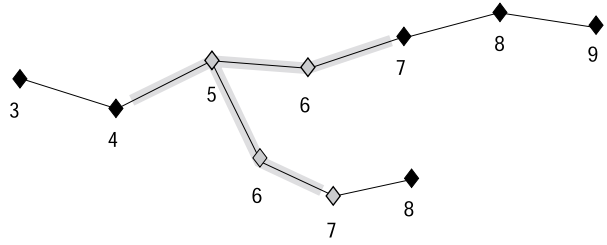


Figure 26. LocateBetween

Pour plus d'informations, reportez-vous à la section «LocateBetween» à la page 179 .

ST_ConvexHull

La fonction ST_ConvexHull renvoie le polygone convexe enveloppe de toutes les géométries dotées d'au moins trois sommets qui forment une convexité. Si les sommets d'une géométrie ne forment pas une convexité, ST_ConvexHull renvoie une valeur NULL. ST_ConvexHull constitue souvent la première étape d'une mosaïque utilisée pour créer un réseau TIN à partir d'un ensemble de points.

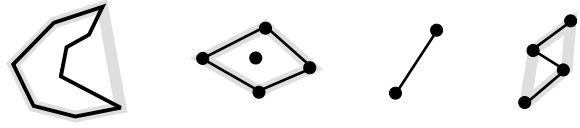


Figure 27. ST_ConvexHull

Pour plus d'informations, reportez-vous à la section «ST_ConvexHull» à la page 204.

ST_Polygon

La fonction ST_Polygon génère un polygone à partir d'une ligne. Pour plus d'informations, reportez-vous à la section «ST_Polygon» à la page 277.

Fonction de conversion du format des valeurs d'une géométrie

DB2 Extension Spatiale prend en charge trois formats d'échange de données SIG :

- Représentation WKT (Well-Known Text)
- Représentation WKB (Well-Known Binary)
- Réprésentation de formes binaire ESRI

Représentation WKT (Well-Known Text)

DB2 Extension Spatiale dispose de plusieurs fonctions de génération de géométries à partir de descriptions textuelles.

ST_WKTToSQL
Crée une géométrie à partir de la représentation textuelle de tout type de géométrie. Vous ne devez spécifier aucun ID de système de références spatiales. Pour plus d'informations, reportez-vous à la section «ST_WKTToSQL» à la page 290.

ST_GeomFromText
Crée une géométrie à partir de la représentation textuelle de tout type de géométrie. Vous devez spécifier un ID de système de références spatiales. Pour plus d'informations, reportez-vous à la section «ST_GeometryFromText» à la page 222.

ST_PointFromText
Crée un point à partir de la représentation textuelle d'un point. Pour plus d'informations, reportez-vous à la section «ST_PointFromText» à la page 268.

ST_LineFromText
Crée une ligne à partir de la représentation textuelle d'une ligne. Pour plus d'informations, reportez-vous à la section «ST_LineFromText» à la page 249.

ST_PolyFromText
Crée un polygone à partir de la représentation textuelle d'un polygone. Pour plus d'informations, reportez-vous à la section «ST_PolyFromText» à la page 274.

ST_MPointFromText
Crée un multipoint à partir de la représentation d'un multipoint. Pour plus d'informations, reportez-vous à la section «ST_MPointFromText» à la page 255.

ST_MLineFromText
Crée un multiligne à partir de la représentation d'une multiligne. Pour plus d'informations, reportez-vous à la section «ST_MLineFromText» à la page 252.

ST_MPolyFromText
Crée un multipolygone à partir de la représentation d'un multipolygone. Pour plus d'informations, reportez-vous à la section «ST_MPolyFromText» à la page 258.

La représentation textuelle consiste en une chaîne ASCII. Elle permet d'échanger une géométrie en format de texte ASCII. Ces fonctions ne nécessitent pas de définir des structures de programme spécifiques pour le mappage des représentations binaires. Par conséquent, elles peuvent être utilisées dans un programme conçu dans un langage de troisième ou de quatrième génération.

La fonction ST_AsText convertit une valeur de géométrie existante en une représentation textuelle. Pour plus d'informations, reportez-vous à la section «ST_AsText» à la page 196.

Pour une description détaillée des représentations textuelles connues (WKT - well-known text), reportez-vous à la section «Représentations textuelles connues (WK)T de l'OGC» à la page 307.

Représentation WKB (Well-Known Binary)

DB2 Extension Spatiale dispose de plusieurs fonctions de génération de géométries à partir de représentations de type WKB (Well-Known Binary).

ST_WKBToSQL
Crée une géométrie à partir d'une représentation binaire connue (WKB) de tout type de géométrie. Vous ne devez spécifier aucun ID de système de références spatiales. Pour plus d'informations, reportez-vous à la section «ST_WKBToSQL» à la page 288.

ST_GeomFromWKB
Crée une géométrie à partir d'une représentation binaire connue (WKB) de tout type de géométrie. Vous devez spécifier un ID de système de références spatiales. Pour plus d'informations, reportez-vous à la section «ST_GeomFromWKB» à la page 224.

ST_PointFromWKB
Crée un point à partir d'une représentation binaire connue (WKB) d'un point. Pour plus d'informations, reportez-vous à la section «ST_PointFromWKB» à la page 269.

ST_LineFromWKB
Crée une ligne à partir d'une représentation binaire connue (WKB) d'une ligne. Pour plus d'informations, reportez-vous à la section «ST_LineFromWKB» à la page 250.

ST_PolyFromWKB
Crée un polygone à partir d'une représentation binaire connue (WKB) d'un polygone. Pour plus d'informations, reportez-vous à la section «ST_PolyFromWKB» à la page 275.

ST_MPointFromWKB
Crée un multipoint à partir d'une représentation binaire connue (WKB) d'un multipoint. Pour plus d'informations, reportez-vous à la section «ST_MPointFromWKB» à la page 256.

ST_MLineFromWKB
Crée une multiligne à partir d'une représentation binaire connue (WKB) d'une multiline. Pour plus d'informations, reportez-vous à la section «ST_MLineFromWKB» à la page 253.

ST_MPolyFromWKB
Crée un multipolygone à partir d'une représentation binaire connue (WKB) d'un multipolygone. Pour plus d'informations, reportez-vous à la section «ST_MPolyFromWKB» à la page 259.

La représentation binaire connue est un flot contigu d'octets. Elle permet d'échanger une géométrie entre un client ODBC et une base de données SQL sous une forme binaire. Ces fonctions impliquent de définir des structures C pour le mappage de la représentation binaire. Par conséquent, elles sont destinées à être utilisées avec un programme écrit en langage de troisième génération et ne conviennent pas à un environnement de langage de quatrième génération.

La fonction ST_AsBinary convertit une valeur de géométrie existante en une représentation binaire connue (WKB). Pour plus d'informations, reportez-vous à la section «ST_AsBinary» à la page 195.

Pour une description détaillée des représentations binaires connues (WKT - well-known text), reportez-vous à la section «Représentations binaires connues (WKB - well-known binary) de l'OGC» à la page 312.

Représentation de formes ESRI

DB2 Extension Spatiale dispose de plusieurs fonctions de génération de géométries à partir de représentations de formes ESRI. La représentation de formes ESRI prend en charge les coordonnées Z et les mesures en sus des représentations bidimensionnelles prises en charge par les représentations textuelles et binaires connues.

ShapeToSQL
Crée une géométrie à partir de la forme de tout type de géométrie. Vous ne devez spécifier aucun ID de système de références spatiales. Pour plus d'informations, reportez-vous à la section «ShapeToSQL» à la page 191.

GeometryFromShape
Crée une géométrie à partir de la forme de tout type de géométrie. Vous devez spécifier un ID de système de références spatiales. Pour plus d'informations, reportez-vous à la section «GeometryFromShape» à la page 169.

PointFromShape
Crée un point à partir de la forme d'un point. Pour plus d'informations, reportez-vous à la section «PointFromShape» à la page 187.

LineFromShape
Crée une ligne à partir de la forme d'une polyligne. Pour plus d'informations, reportez-vous à la section «LineFromShape» à la page 175.

PolyFromShape
Crée un polygone à partir de la forme d'une polyligne. Pour plus d'informations, reportez-vous à la section «PolyFromShape» à la page 189.

MPointFromShape
Crée un multipoint à partir de la forme d'un multipoint. Pour plus d'informations, reportez-vous à la section «MPointFromShape» à la page 184.

MLineFromShape
Crée une multiligne à partir de la forme d'une multiligne. Pour plus d'informations, reportez-vous à la section «MLineFromShape» à la page 182.

MPolyFromShape
Crée un multipolygone à partir de la forme d'un multipolygone. Pour plus d'informations, reportez-vous à la section «MPolyFromShape» à la page 186.

La syntaxe générale de ces fonctions est identique. Le premier argument est la représentation de la forme entrée en tant que type de données BLOB. Le second argument est l'ID de référence spatiale qui doit être affecté à la géométrie. Ainsi, la fonction GeometryFromShape doit respecter la syntaxe suivante :

```
GeometryFromShape(shapegeometry, SRID)
```

Pour mapper la représentation binaire, ces fonctions de forme nécessitent de définir des structures C. Par conséquent, elles sont destinées à être utilisées avec un programme écrit en langage de troisième génération et ne conviennent pas à un environnement de langage de quatrième génération.

La fonction AsBinaryShape convertit la valeur d'une géométrie en une représentation de forme ESRI. Pour plus d'informations, reportez-vous à la section «AsBinaryShape» à la page 168.

Pour une description détaillée des représentations de formes, reportez-vous à la section «Représentation de forme ESRI» à la page 316.

Chapitre 14. Fonctions spatiales associées aux requêtes SQL

Le présent chapitre répertorie les fonctions que vous pouvez appeler pour analyser des données spatiales. Chaque fonction est décrite dans une section qui illustre la syntaxe, le code de retour et comporte des exemples de code. Certains exemples fournis dans ce chapitre comportent une instruction CREATE TABLE dans laquelle une voire plusieurs colonnes sont définies en tant que colonnes spatiales.

Les considérations suivantes s'appliquent aux fonctions spatiales :

- Les exemples décrits dans ce chapitre sont qualifiés avec le nom de bibliothèque db2gse. Au lieu de qualifier explicitement chaque fonction et type spatiaux avec ce nom, vous pouvez définir le chemin de fonction de manière à ce qu'il inclut db2gse.
- Avant d'insérer des données dans une colonne spatiale :
  - Vous devrez peut-être augmenter la valeur du paramètre udf_mem_sz. La valeur initiale suggérée est de 2048. Si elle ne convient pas, augmentez le paramètre udf_mem_sz par incréments de 256.
  - Vous devez enregistrer la colonne en tant que couche. Pour plus d'informations sur l'enregistrement d'une colonne spatiale en tant que couche, reportez-vous au «Chapitre 4. Définition et enregistrement des colonnes spatiales, et activation d'un géocodeur» à la page 33.

AsBinaryShape

AsBinaryShape utilise un objet de type géométrie en tant qu'entrée et renvoie un objet de type BLOB.

Syntaxe

db2gse.AsBinaryShape(g db2gse.ST_Geometry)

Type de retour

BLOB(1m)

Exemples

Le fragment de code présenté ci-après illustre comment la fonction AsBinaryShape convertit les polygones de zone de la table SENSITIVE_AREAS en polygones de forme. Ces polygones de type forme sont transmis à la fonction draw_polygon de l'application à des fins d'affichage.

```
/* Create the SQL expression. */
strcpy(sqlstmt, "select db2gse.AsBinaryShape (zone) from SENSITIVE_AREAS
where db2gse.EnvelopesIntersect(zone, db2gse.PolyFromShape(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Prepare the SQL statement. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* Set the pcbvalue1 length of the shape. */
pcbvalue1 = blob_len;

/* Bind the shape parameter */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB, blob_len,
0, shape, blob_len, &pcbvalue1);

/* Execute the query */
rc = SQLExecute (hstmt);

/* Assign the results of the query (the Zone polygons) to the
   fetched_binary variable. */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* Fetch each polygon within the display window and display it. */

while(SQL_SUCCESS == (rc = SQLFetch(hstmt)))
  draw_polygon(fetched_binary);
```

GeometryFromShape

GeometryFromShape utilise une forme et une identité de système de références spatiales en tant qu'entrée, et renvoie un objet de type géométrie.

Syntaxe

db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)

Type de retour

db2gse.ST_Geometry

Exemples

Le fragment de code C présenté ci-dessous contient des fonctions ODBC imbriquées dans des fonctions SQL DB2 Extension Spatiale qui insèrent des données dans la table LOTS.

La table LOTS a été créée avec deux colonnes : la colonne LOT_ID qui identifie chaque parcelle de façon univoque et la colonne des polygones LOT, qui contient la géométrie de chaque parcelle.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot     db2gse.ST_MultiPolygon);
```

La fonction GeometryFromShape convertit les formes en géométrie DB2 Extension Spatiale. La totalité de l'instruction INSERT est copiée dans shp_sql. L'instruction INSERT contient des marqueurs de paramètre permettant d'accepter dynamiquement les données LOT_ID et LOT.

```
/* Create the SQL insert statement to populate the lot id and the
   lot multipolygon. The question marks are parameter markers that
   indicate the lot_id and lot values that will be retrieved at
   runtime. */
strcpy (shp_sql,"insert into LOTS (lot_id,
lot) values (?,
db2gse.GeometryFromShape (cast(? as
blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Bind the integer key value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Bind the shape to the second parameter. */
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

EnvelopesIntersect

EnvelopesIntersect renvoie la valeur 1 (TRUE) si les enveloppes de deux géométries génèrent une intersection et la valeur 0 (FALSE), dans le cas contraire.

Syntaxe

db2gse.EnvelopesIntersect(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

La fonction get_window extrait les coordonnées de la fenêtre d'affichage à partir de l'application. Le paramètre de la fenêtre est en l'occurrence une structure de forme polygonale contenant une chaîne de coordonnées qui représentent le polygone d'affichage. La fonction PolyFromShape convertit la forme de la fenêtre d'affichage en un polygone DB2 Extension Spatiale ; la fonction EnvelopesIntersect se sert de celui-ci en tant qu'enveloppe de l'intersection qu'elle génère. Tous les polygones de zone SENSITIVE_AREAS qui forment une intersection avec l'intérieur ou le contour de la fenêtre d'affichage sont renvoyés. Chaque polygone est extrait de l'ensemble de résultats et transmis à la fonction draw_polygon.

```
/* Get the display window coordinates as a polygon shape.
get_window(&window)

/* Create the SQL expression. The db2gse.EnvelopesIntersect function
  will be used to limit the result set to only those zone polygons
  that intersect the envelope of the display window. */
strcpy(sqlstmt, "select db2gse.AsBinaryShape(zone) from SENSITIVE_AREAS where
db2gse.EnvelopesIntersect (zone,
db2gse.PolyFromShape(cast(? as
blob(1m)),
db2gse.coordref()..srid(0)))");

/* Set blob_len to the byte length of a 5 point shape polygon. */
blob_len = 128;

/* Prepare the SQL statement. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* Set the pcbvalue1 to the window shape */
pcbvalue1 = blob_len;

/* Bind the shape parameter */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT,
SQL_C_BINARY, SQL_BLOB,
blob_len,0, window, blob_len, &pcbvalue1);

/* Execute the query */
rc = SQLExecute (hstmt);
```

```
/* Assign the results of the query, (the Zone polygons) to the
   fetched_binary variable. */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* Fetch each polygon within the display window and display it. */
while(SQL_SUCCESS == (rc = SQLFetch(hstmt))
  draw_polygon(fetched_binary);
```

Is3d

Is3d utilise un objet de type géométrie en tant qu'entrée et renvoie la valeur 1 (TRUE) s'il a des coordonnées tridimensionnelles, et la valeur 0 (FALSE), si tel n'est pas le cas.

Syntaxe

db2gse.Is3d(g db2gse.ST_Geometry)

Type de retour

Integer

Exemples

L'instruction CREATE TABLE ci-après crée la table THREED_TEST qui comporte deux colonnes : la colonne GID de type entier et la colonne G1 de type géométrie.

```
CREATE TABLE THREED_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Les instructions INSERT insèrent deux points dans la table TREED_TEST table. Le premier point ne comporte pas de coordonnées Z alors que le second en a.

```
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)))
```

L'instruction SELECT ci-dessous affiche le contenu de la colonne GID accompagné des résultats de la fonction Is3d. La fonction renvoie la valeur 0 pour la première ligne de la table sans coordonnées et la valeur 1 pour la seconde qui est dotée de coordonnées Z.

```
SELECT gid, db2gse.Is3d (g1) "Is it 3d?" FROM THREED_TEST
```

L'ensemble de résultats suivant est renvoyé :

```
gid      Is it 3d?
------   ----------
     1            0
     2            1
```

IsMeasured

IsMeasured utilise un objet de type géométrie en tant qu'entrée et renvoie la valeur 1 (TRUE) s'il est doté de mesures et la valeur 0 (FALSE) dans le cas contraire.

Syntaxe

db2gse.IsMeasured(g db2gse.ST_Geometry)

Type de retour

Integer

Exemples

L'instruction CREATE TABLE ci-après crée la table MEASURE_TEST, qui comporte deux colonnes. La colonne GID identifie les lignes de la table de manière univoque et la colonne G1 sert à stocker les géométries de type point.

```
CREATE TABLE MEASURE_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Les instructions INSERT présentées ci-dessous insèrent deux enregistrements dans la table MEASURE_TEST. Le premier stocke un point qui n'a pas de mesure, le second, un point doté d'une mesure.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant présentent la colonne GID accompagnée du résultat de la fonction IsMeasured. La fonction IsMeasured renvoie la valeur 0 pour la première ligne de la table car le point n'a pas de mesure, et la valeur 1 pour la seconde ligne parce que le point est doté d'une mesure.

```
SELECT gid, db2gse.IsMeasured (g1) "Has measures?" FROM MEASURE_TEST
gid      Has measures
------   ----------
     1            0
     2            1
```

LineFromShape

LineFromShape utilise une forme de type point et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie une ligne.

Syntaxe

db2gse.LineFromShape(ShapeLineString Blob(1M), cr db2gse.coordref)

Type de retour

db2gse.ST_LineString

Exemples

Le fragment de code ci-après peuple la table SEWERLINES avec l'ID unique, la classe de taille et la géométrie de chaque canalisation d'égout.

L'instruction CREATE TABLE présentée ci-après crée la table SEWERLINES qui comporte trois colonnes. La première, SEWER_ID, identifie de manière univoque chaque canalisation. La seconde, CLASS, de type entier identifie le type de canalisation d'égout qui est généralement associé à la capacité de la canalisation. La troisième colonne, SEWER, de type ligne stocke la géométrie de la canalisation d'égout.

```
CREATE TABLE SEWERLINES (sewer_id   integer, class  integer,
                         sewer db2gse.ST_LineString);

/* Create the SQL insert statement to populate the sewer_id, size class and
   the sewer linestring. The question marks are parameter markers that
   indicate the sewer_id, class and sewer geometry values that will be
   retrieved at runtime. */
strcpy (shp_sql,"insert into sewerlines (sewer_id,class,sewer)
values (?,?, db2gse.Line FromShape
(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Bind the integer key value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_id, 0, &pcbvalue1);

/* Bind the integer class value to the second parameter. */
pcbvalue2 = 0;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_class, 0, &pcbvalue2);;

/* Bind the shape to the third parameter. */
pcbvalue3 = blob_len;
```

```
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, sewer_shape, blob_len, &pcbvalue3);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

LocateAlong

LocateAlong utilise un objet de type géométrie et une mesure en tant qu'entrée, et renvoie l'ensemble de points trouvés à la mesure sous forme de multipoint.

Syntaxe

db2gse.LocateAlong(g db2gse.ST_Geometry, adistance Double)

Type de retour

db2gse.ST_Geometry

Exemples

L'instruction CREATE TABLE ci-après crée la table LOCATEALONG_TEST qui comporte deux colonnes : la colonne GID qui identifie chaque ligne de la table de manière univoque et la colonne de type géométrie G1 qui stocke la géométrie exemple.

```
CREATE TABLE LOCATEALONG_TEST (gid integer, g1 db2gse.ST_Geometry)
```

Les instructions INSERT ci-après insèrent deux lignes de table. La première est une multiligne, et la seconde, un multipoint.

```
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)))
```

Dans l'instruction SELECT ci-après et l'ensemble de résultats correspondant, la fonction LocateAlong est conçue pour détecter les points qui ont une mesure de 6,5. La première ligne de la table renvoie un multipoint contenant deux points. Par contre, la seconde renvoie un point vide. Pour les entités linéaires (géométries dotées d'une dimension supérieure à 0), LocateAlong peut interpoler le point ; par contre, dans le cas de multipoints, la mesure cible doit être exactement identique.

```
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.
```

Dans l'instruction SELECT ci-après et l'ensemble de résultats correspondant, la fonction LocateAlong renvoie des multipoints pour les deux lignes de la table. La mesure cible de valeur 7 est exactement identique aux mesures contenues dans les données source de la multiligne et du multipoint.

```
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

LocateBetween utilise un objet de type géométrie et deux positions de mesure en tant qu'entrée, et renvoie une géométrie qui représente l'ensemble des chemins disconnectés contenus entre deux positions de mesure.

Syntaxe

db2gse.LocateBetween(g db2gse.ST_Geometry, adistance Double, anotherdistance Double)

Type de retour

db2gse.ST_Geometry

Exemples

L'instruction CREATE TABLE ci-après crée la table LOCATEBETWEEN_TEST qui comporte deux colonnes : la colonne GID qui identifie chaque ligne de la table de manière univoque et la colonne de type géométrie G1 qui stocke la géométrie exemple.

```
CREATE TABLE LOCATEBETWEEN_TEST (gid integer, g1 db2gse.ST_Geometry)
```

Les instructions INSERT présentées ci-dessous insèrent deux lignes dans la table LOCATEBETWEEN_TEST. La première est une multiligne, et la seconde, un multipoint.

```
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)))
```

L'instruction SELECT et l'ensemble de résultats correspondant ci-après indiquent comment la fonction LocateBetween localise les mesures figurant entre les valeurs 6,5 et 7,5 incluses. La première ligne de la table renvoie une multiligne composée de plusieurs lignes, et la seconde, un multipoint car les données source étaient de ce type. Lorsque les données source ont une dimension de valeur 0 (point ou multipoint), la fonction exige une correspondance exacte.

```
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 utilise un point en tant qu'entré et renvoie une mesure.

Syntaxe

db2gse.M(p db2gse.ST_Point)

Type de retour

Double

Exemples

L'instruction CREATE TABLE ci-après crée la table M_TEST qui comporte deux colonnes : la colonne GID qui identifie chaque ligne de la table de manière univoque et la colonne de type point PT1 qui stocke la géométrie exemple.

```
CREATE TABLE M_TEST (gid integer, pt1 db2gse.ST_Point)
```

Les instructions INSERT ci-après insèrent une ligne contenant un point doté de mesures et une ligne contenant un point sans mesures.

```
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)))
```

Dans l'instruction SELECT ci-après et l'ensemble de résultats correspondant, la fonction M répertorie les valeurs de mesure des points. Le premier n'ayant pas de mesure, la fonction M renvoie une valeur NULL.

```
SELECT gid, db2gse.M (pt1) "The measure" FROM M_TEST

GID         The measure
----------- ------------------------
          1                        -
          2   +7.00000000000000E+000

  2 record(s) selected.
```

MLineFromShape

MLineFromShape utilise une forme de type multiligne et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie une multiligne.

Syntaxe

db2gse.MLineFromShape(ShapeMultiLineString Blob(1M), cr db2gse.coordref)

Type de retour

db2gse.ST_MultiLineString

Exemples

Le fragment de code ci-après peuple la table WATERWAYS avec un ID unique, un nom et une multiligne.

La table WATERWAYS est créée avec les colonnes ID et NAME, qui identifient chaque système hydrographique (rivières et ruisseaux) contenu dans la table. La colonne WATER est de type multiligne car ces systèmes sont souvent un agrégat de plusieurs lignes.

```
CREATE TABLE WATERWAYS (id        integer,
                        name      varchar(128),
                        water     db2gse.ST_MultiLineString);

/* Create the SQL insert statement to populate the id, name and
   multilinestring. The question marks are parameter markers that
   indicate the id, name and water values that will be retrieved at
   runtime. */
strcpy (shp_sql,"insert into WATERWAYS (id,name,water)
values (?,?, db2gse.MLineFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Bind the integer id value to the first parameter. */

pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);
/* Bind the varchar name value to the second parameter. */

pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, name_len, 0, &name, name_len, &pcbvalue2);

/* Bind the shape to the third parameter. */
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, blob_len, 0, water_shape, blob_len, &pcbvalue3);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

MPointFromShape

MPointFromShape utilise une forme de type multipoint et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un multipoint.

Syntaxe

db2gse.MPointFromShape(ShapeMultiPoint (1M), srs db2gse.coordref)

Type de retour

db2gse.ST_MultiPoint

Exemples

Le fragment de code ci-après peuple la table SPECIES_SITINGS d'un biologiste.

La table SPECIES_SITINGS est créée avec trois colonnes. Les colonnes SPECIES et GENUS identifient de manière univoque chaque ligne de la table alors que le multipoint de la colonne SITINGS représente les lieux d'implantation de l'espèce.

```
CREATE TABLE SPECIES_SITINGS (species  varchar(32),
                              genus varchar(32),
                              sitings  db2gse.ST_MultiPoint);

/* Create the SQL insert statement to populate the species, genus and
   sitings. The question marks are parameter markers that indicate the
   name and water values that will be retrieved at runtime. */
strcpy (shp_sql,"insert into SPECIES_SITINGS (species,genus,sitings)
values (?,?, db2gse.MPointFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Bind the varchar species value to the first parameter. */
pcbvalue1 = species_len;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, species_len, 0, species, species_len, &pcbvalue1);

/* Bind the varchar genus value to the second parameter. */
pcbvalue2 = genus_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, genus_len, 0, name, genus_len, &pcbvalue2);

/* Bind the shape to the third parameter. */
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, sitings_len, 0, sitings_shape, sitings_len, &pcbvalue3);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

MPolyFromShape

MPolyFromShape utilise une forme de type multipolygone et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un multipolygone.

Syntaxe

db2gse.MPolyFromShape(ShapeMultiPolygon Blob(1m), srs db2gse.coordref)

Type de retour

db2gse.ST_MultiPolygon

Exemples

Le fragment de code ci-après peuple la table LOTS.

La table LOTS stocke l'ID parcelle (lot_id) qui identifie chaque parcelle de manière univoque, ainsi que le multipolygone de parcelle, qui contient la géométrie de type ligne de la dite parcelle.

```
CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );

/* Create the SQL insert statement to populate the lot_id and lot. The
   question marks are parameter markers that indicate the lot_id and lot
   values that will be retrieved at runtime. */
strcpy (shp_sql,"insert into LOTS (lot_id,lot)
values (?, db2gse.MPolyFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Bind the lot_id integer value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Bind the lot shape to the second parameter. */
pcbvalue2 = lot_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, lot_len, 0, lot_shape, lot_len, &pcbvalue2);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

PointFromShape

PointFromShape utilise une forme de type point et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un point.

Syntaxe

db2gse.PointFromShape(db2gse.ShapePoint blob(1M), srs db2gse.coordref)

Type de retour

db2gse.ST_Point

Exemples

Ce fragment de programme peuple la table HAZARDOUS_SITES.

Les sites à risque sont stockés dans la table HAZARDOUS_SITES créée avec l'instruction CREATE TABLE présentée ci-après. La colonne LOCATION, définie en tant que point, stocke une position représentant le centre géographique du site à risque.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);

/* Create the SQL insert statement to populate the site_id, name and
   location. The question marks are parameter markers that indicate the
   site_id, name and location values that will be retrieved at runtime. */
strcpy (shp_sql,"insert into HAZARDOUS_SITES (site_id, name, location)
values (?,?, db2gse.PointFromShape
(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Bind the site_id integer value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);

/* Bind the name varchar value to the second parameter. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Bind the location shape to the third parameter. */
pcbvalue3 = location_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
      SQL_BLOB, location_len, 0, location_shape, location_len, &pcbvalue3);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

PolyFromShape

PolyFromShape utilise une forme de type polygone et une identité de système de références spatiales en tant qu'entrée, et renvoie un polygone.

Syntaxe

db2gse.PolyFromShape (ShapePolygon Blob(1M), srs db2gse.coordref)

Type de retour

db2gse.ST_Polygon

Exemples

Ce fragment de programme peuple la table SENSITIVE_AREAS. Les points d'interrogation représentent les marqueurs de paramètre associés aux valeurs ID, nom, superficie et zone, qui seront extraites au moment de l'exécution.

La table SENSITIVE_AREAS contient plusieurs colonnes qui décrivent les établissements menacés, en sus de la colonne zone qui stocke la géométrie de type polygone des établissements.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);

/* Create the SQL insert statement to populate the id, name, size, type and
   zone. The question marks are parameter markers that indicate the
   id, name, size, type and zone values that will be retrieved at runtime. */
strcpy (shp_sql,"insert into SENSITIVE_AREAS (id, name, size, type, zone)
values (?,?,?,?, db2gse.PolyFromShape
(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Bind the id integer value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);
/* Bind the name varchar value to the second parameter. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Bind the size float to the third parameter. */
pcbvalue3 = 0;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_FLOAT,
```

```
     SQL_REAL, 0, 0, &size, 0, &pcbvalue3);

/* Bind the type varchar to the fourth parameter. */
pcbvalue4 = type_len;
rc = SQLBindParameter (hstmt, 4, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_VARCHAR, type_len, 0, type, type_len, &pcbvalue4);

/* Bind the zone polygon to the fifth parameter. */
pcbvalue5 = zone_len;
rc = SQLBindParameter (hstmt, 5, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, zone_len, 0, zone_shp, zone_len, &pcbvalue5);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

ShapeToSQL

ShapeToSQL crée une valeur db2gse.ST_Geometry en fonction de sa représentation binaire connue (WKB). La valeur SRID 0 est automatiquement utilisée.

Syntaxe

db2gse.ST_ShapeToSQL(ShapeGeometry blob(1M))

Type de retour

db2gse.ST_Geometry

Exemples

Le fragment de code C présenté ci-dessous contient des fonctions ODBC imbriquées dans des fonctions SQL DB2 Extension Spatiale qui insèrent des données dans la table LOTS. La table LOTS a été créée avec deux colonnes : la colonne lot_id qui identifie chaque parcelle de façon univoque, et la colonne des multipolygones de parcelle, qui contient la géométrie de chaque parcelle.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

La fonction ShapeToSQL convertit les formes en géométrie DB2 Extension Spatiale. La totalité de l'instruction INSERT est copiée dans shp_sql. L'instruction INSERT contient des marqueurs de paramètre permettant d'accepter dynamiquement l'ID parcelle (ID_lot) et les données sur la parcelle.

```
/* Create the SQL insert statement to populate the lot id and the
   lot multipolygon. The question marks are parameter markers that
   indicate the lot_id and lot values that will be retrieved at
   run time. */

strcpy (shp_sql,"insert into lots (lot_id, lot) values(?,
db2gse.ShapeToSQL(cast(? as blob(1m))))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */

rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */

rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Bind the integer key value to the first parameter. */

pcbvalue1 = 0;

rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Bind the shape to the second parameter. */
```

```
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Execute the insert statement. */

rc = SQLExecute (hstmt);
```

ST_Area

ST_Area utilise un polygone ou un multipolygone en tant qu'entrée et renvoie la surface correspondante.

Syntaxe

db2gse.ST_Area(s db2gse.ST_Surface)

Type de retour

Double

Exemples

Le directeur des services techniques municipaux a besoin de la liste des surfaces bâties. Pour l'obtenir, un technicien SIG sélectionne l'ID (building_id) et la surface de chaque bâti.

Les bâtis sont enregistrés dans la table BUILDINGFOOTPRINTS créée avec l'instruction CREATE TABLE :

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

Pour répondre à la demande du directeur des services techniques, le technicien a utilisé l'instruction SELECT ci-après pour sélectionner la clé unique, l'ID bâtiment (id_building), et la surface de chaque bâti figurant dans la table BUILDINGFOOTPRINTS.

```
SELECT building_id, db2gse.ST_Area (footprint) "Area"
FROM BUILDINGFOOTPRINTS;
```

L'instruction SELECT renvoie l'ensemble de résultats suivant :

```
building_id   Area
------------  ------------------------
         506    +1.40768000000000E+003
        1208    +2.55759000000000E+003
         543    +1.80786000000000E+003
         178    +2.08671000000000E+003
         .
         .
         .
```

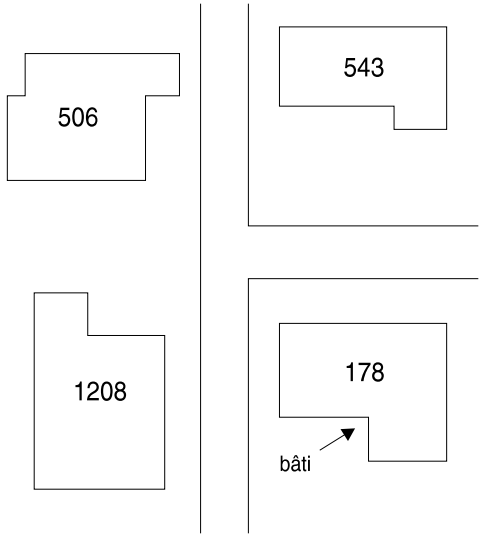


Figure 28. Extraction d'un bâti à partir de la surface. Quatre bâtis libellés par leur numéros d'ID bâtiment sont représentés le long de la rue adjacente correspondante.

ST_AsBinary

ST_AsBinary utilise un objet de type géométrie en tant qu'entrée et renvoie une représentation binaire connue (WKB - well-known binary).

Syntaxe

db2gse.ST_AsBinary(g db2gse.ST_Geometry)

Type de retour

BLOB(1m)

Exemples

Le fragment de code suivant illustre comment la fonction ST_AsBinary convertit les multipolygones des bâtis contenus dans la table BUILDINGFOOTPRINTS en multipolygones de type WKB. Ceux-ci sont transmis à la fonction draw_polygon à des fins d'affichage.

```
/* Create the SQL expression. */
strcpy(sqlstmt, "select db2gse.ST_AsBinary (footprint) from BUILDINGFOOTPRINTS
where db2gse.EnvelopesIntersect(footprint,
db2gse.ST_PolyFromWKB(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Prepare the SQL statement. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* Set the pcbvalue1 length of the shape. */
pcbvalue1 = blob_len;

/* Bind the shape parameter */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB, blob_len,
0, shape, blob_len, &pcbvalue1);

/* Execute the query */
rc = SQLExecute (hstmt);

/* Assign the results of the query (the Zone polygons) to the
  fetched_binary variable.
 */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* Fetch each polygon within the display window and display it. */
while(SQL_SUCCESS == (rc = SQLFetch(hstmt)))
  draw_polygon(fetched_binary);
```

ST_AsText

db2gse.ST_AsText utilise un objet de type géométrie en tant qu'entrée et renvoie une représentation binaire connue (WKB - well-known text).

Syntaxe

db2gse.ST_AsText(g db2gse.ST_Geometry)

Type de retour

Varchar(4000)

Exemples

Dans le scénario présenté ci-après, la fonction db2gse.ST_AsText convertit le point de l'emplacement HAZARDOUS_SITES en la description textuelle correspondante :

```
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;
```

L'instruction SELECT renvoie l'ensemble de résultats suivant :

```
SITE_ID Name                                Location
------- ----------------------------------- ---------------------------------
    102 W. H. Kleenare Chemical Repository  POINT (1020.00000000 324.00000000)
```

ST_Boundary

ST_Boundary utilise un objet de type géométrie en tant qu'entrée et renvoie le contour combiné correspondant sous forme d'objet de type géométrie.

Syntaxe

db2gse.ST_Boundary(g db2gse.ST_Geometry)

Type de retour

db2gse.ST_Geometry

Exemples

Dans le fragment de code ci-après, la table BOUNDARY_TEST est créée. Elle comporte deux colonnes : GEOTYPE, qui est défini en tant que type de données varchar, et G1, défini comme la géométrie de superclasse. Les instructions INSERT ci-dessous insèrent chacune des géométries appartenant aux sous-classes. La fonction ST_Boundary extrait le contour de chaque sous-classe qui est stocké dans la colonne de géométrie G1. La dimension de la géométrie obtenue est toujours inférieure d'une unité à celle de la géométrie source. Le résultat des points et des multipoints est toujours un contour consistant en une géométrie vide de dimension 1. Les lignes et les multilignes renvoient un contour de type multipoint de dimension 0, et les polygones et multipolygones, un contour de type multiligne de dimension 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 utilise un objet de type géométrie et une distance en tant qu'entrée, et renvoie la géométrie qui entoure l'objet source.

Syntaxe

db2gse.ST_Buffer(g db2gse.ST_Geometry, adistance Double)

Type de retour

db2gse.ST_Geometry

Exemples

L'administrateur du comté a besoin de la liste des sites à risque qui chevauchent des zones sensibles telles que les écoles, les hôpitaux et les maisons de retraite. Les zones sensibles sont enregistrées dans la table SENSITIVE_AREAS créée avec l'instruction CREATE TABLE ci-dessous. La colonne ZONE est définie en tant que polygone, qui est stocké sous la forme du contour de chaque zone sensible.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

Les sites à risque sont stockés dans la table HAZARDOUS_SITES créée avec l'instruction CREATE TABLE présentée ci-après. La colonne LOCATION, définie en tant que point, stocke une position représentant le centre géographique de chaque site à risque.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

Les tables SENSITIVE_AREAS et HAZARDOUS_SITES sont jointes à l'aide de la fonction db2gse.ST_Overlaps. La fonction renvoie la valeur 1 (TRUE) pour toutes les lignes de la table SENSITIVE_AREAS dont les polygones de surface chevauchent le périmètre tampon d'un rayon d'environ 7 km défini autour du point représentant l'emplacement 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;
```

Sur la figure 29 à la page 200, certaines des zones sensibles de cette division administrative sont situées à l'intérieur du périmètre tampon de 7 km défini autour des emplacements des sites à risque. Les deux zones tampons forment une intersection avec l'hôpital et l'une d'elle avec l'école. Par contre, la maison de retraite est située en dehors des deux périmètres de sécurité.

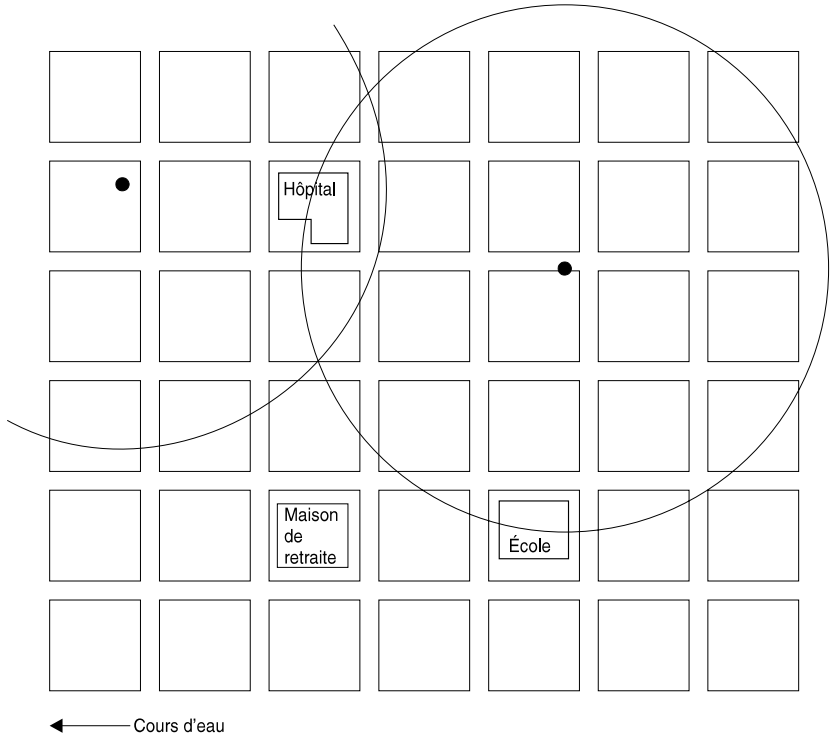


Figure 29. Une zone tampon d'un rayon d'environ 7 km est tracée autour d'un point

ST_Centroid

ST_Centroid utilise un polygone ou un multipolygone en tant qu'entrée et renvoie le centre géométrique (centroïde) correspondant sous forme de point.

Syntaxe

db2gse.ST_Centroid(s db2gse.ST_Surface)
db2gse.ST_Centroid(ms db2gse.ST_MultiSurface)

Type de retour

Pour une surface : db2gse.ST_Point

Exemples

Le technicien SIG de la ville veut afficher les multipolygones des bâtis sous la forme de points simples dans un graphique représentant le mode d'occupation des sols.

Les bâtis sont enregistrés dans la table BUILDINGFOOTPRINTS créée avec l'instruction CREATE TABLE ci-après.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                         footprint   db2gse.ST_MultiPolygon);
```

La fonction ST_Centroid renvoie le centroïde de chaque multipolygone représentant un bâti. La fonction AsBinaryShape convertit le point centroïde en une forme qui est la représentation externe reconnue par l'application.

```
SELECT building_id,
       CAST(db2gse.AsBinaryShape(db2gse.ST_Centroid (footprint))
       as blob(1m)) "Centroid"
FROM BUILDINGFOOTPRINTS;
```

ST_Contains

ST_Contains utilise deux géométries en tant qu'entrée et renvoie la valeur 1 (TRUE) si le premier objet contient entièrement le second et la valeur 0 (FALSE), si tel n'est pas le cas.

Syntaxe

db2gse.ST_Contains(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

Dans l'exemple ci-dessous, deux tables sont créées. L'une contient le bâti d'une ville, et l'autre, le parcellaire. Le directeur des services techniques municipaux veut s'assurer que tous les bâtis sont entièrement contenus dans leur parcelle respective.

Dans les deux tables, le type de données multipolygone permet de stocker la géométrie des bâtis et celle des parcelles. Le concepteur de la base de données a sélectionné des multipolygones pour ces deux entités. Il a tenu compte du fait que les parcelles pouvaient être disjointes par des entités naturelles (rivière, etc.) et qu'un bâti pouvait souvent être constitué de plusieurs bâtiments.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                         footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

Le directeur des services techniques municipaux a d'abord sélectionné les bâtis qui n'étaient pas entièrement contenus dans une seule parcelle.

```
SELECT building_id
  FROM BUILDINGFOOTPRINTS, LOTS
WHERE db2gse.ST_Contains(lot,footprint) = 0;
```

Ensuite, il a réalisé que la première requête renverrait la liste de tous les ID bâtiment associés à des bâtis figurant en dehors d'un polygone de parcelle. Mais il savait également que ces informations n'indiqueraient pas si l'ID parcelle correct avait été affecté aux autres bâtiments. La seconde requête exécute une vérification de l'intégrité des données sur la colonne lot_id de la table 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;
```

Sur la figure 30, les bâtis libellés avec leurs ID bâtiment se trouvent à l'intérieur de leurs parcelles. Les limites des parcelles sont représentées par des lignes en pointillé. Bien que cela n'apparaisse pas, ces lignes s'étendent jusqu'au milieu de la rue et englobent les parcelles ainsi que les bâtis contenus dans celles-ci.

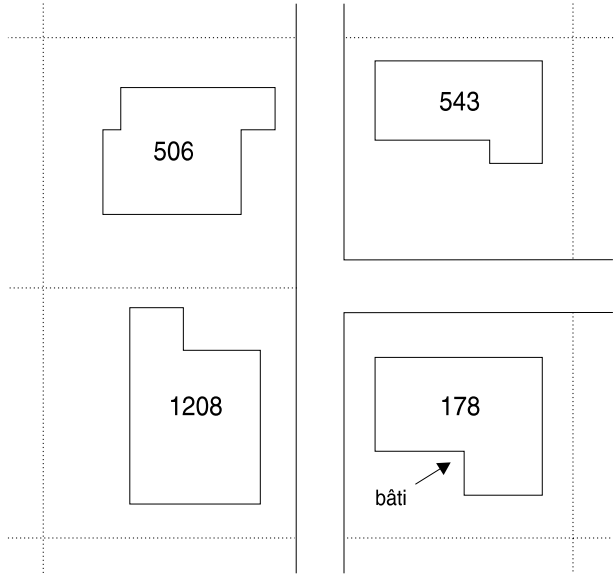


Figure 30. Vérification à l'aide de la fonction ST_Contains que tous les bâtiments sont contenus dans leurs parcelles

ST_ConvexHull

AsShape utilise un objet de type géométrie en tant qu'entrée et renvoie l'enveloppe convexe correspondante.

Syntaxe

db2gse.ST_ConvexHull(g db2gse.ST_Geometry)

Type de retour

db2gse.ST_Geometry

Exemples

L'exemple ci-après crée la table CONVEXHULL_TEST qui comporte deux colonnes : GEOTYPE et G1. La colonne GEOTYPE, type de données varchar(20), permet d'enregistrer le nom de la sous-classe de la géométrie stockée dans la colonne G1, définie en tant que géométrie.

```
CREATE TABLE CONVEXHULL_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Chaque instruction INSERT insère une géométrie de chaque type de sous-classe dans la table 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)))
```

L'instruction SELECT ci-après affiche le nom de sous-classe stocké dans la colonne GEOTYPE et l'enveloppe convexe. L'enveloppe convexe générée par la fonction ST_ConvexHull est convertie en texte par la fonction ST_AsText. Elle est ensuite transtypée en données de varchar(256) car la sortie par défaut de la fonction ST_AsText consiste en des données de type varchar(4000).

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText(db2gse.ST_ConvexHull(G1)))
as varchar(256) "The convexhull"
FROM CONVEXHULL_TEST
```

ST_CoordDim

ST_CoordDim renvoie les dimensions des coordonnées associées à la valeur ST_Geometry. Pour l'explication des dimensions des coordonnées, reportez-vous à la section «Points» à la page 136.

Syntaxe

db2gse.ST_CoordDim(g1 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

La table coorddim_test est crée avec les colonnes GEOTYPE et G1. La colonne GEOTYPE stocke le nom de la sous-classe de la géométrie enregistrée dans la colonne G1.

```
CREATE TABLE coorddim_test (geotype
varchar(20), g1 db2gse.ST_Geometry)
```

Les instructions INSERT insèrent une sous-classe exemple dans la table 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))
)
```

L'instruction SELECT ci-après répertorie le nom de la sous-classe stocké dans la colonne geotype, accompagné des dimensions des coordonnées associées à ce type de géométrie.

```
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 utilise deux objets de type géométrie en tant qu'entrée et renvoie la valeur 1 (TRUE) si leur intersection génère une géométrie dont la dimension est inférieure d'une unité à la dimension maximale des objets source. L'objet intersection contient des points qui figurent à l'intérieur des deux géométries source et il n'est égal à aucun de ces deux objets. Dans le cas contraire, il renvoie la valeur 0 (FALSE).

Syntaxe

db2gse.ST_Crosses(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

Le gouvernement du comté étudie une nouvelle loi stipulant qu'aucune installation de stockage de produits dangereux du comté ne doit se trouver à moins de 7 km de tout cours d'eau. Le gestionnaire SIG du comté dispose d'une représentation précise des cours d'eau (rivières et ruisseaux) qui sont enregistrés sous forme de multilignes dans la table WATERWAYS. Toutefois, la position de chaque installation de stockage des produits dangereux n'est indiquée que par un seul point.

```
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);
```

Pour déterminer s'il convient de signaler les installations contrevenant au projet de loi à l'administrateur du comté, le gestionnaire SIG devrait entourer les emplacements des sites à risque par une zone tampon et constater si des rivières et des ruisseaux traversent le polygone tampon. Le prédicat ST_Crosses compare les sites à risque contenus dans la table HAZARDOUS_SITES et entourés par une zone tampon avec la table WATERWAYS. Ainsi, le prédicat ne renvoie que les enregistrements dans lesquels le cours d'eau coupe le rayon indiqué par le projet de loi.

```
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;
```

Sur la figure 31 à la page 209, le tampon d'un rayon de 7 km créé autour des sites de déchets dangereux coupe le réseau de ruisseaux parcourant la division administrative du comté. Ce réseau de ruisseaux a été défini en tant

que multiligne. Par conséquent, l'ensemble de résultats comprend tous les segments de ligne appartenant aux segments qui coupent le rayon.

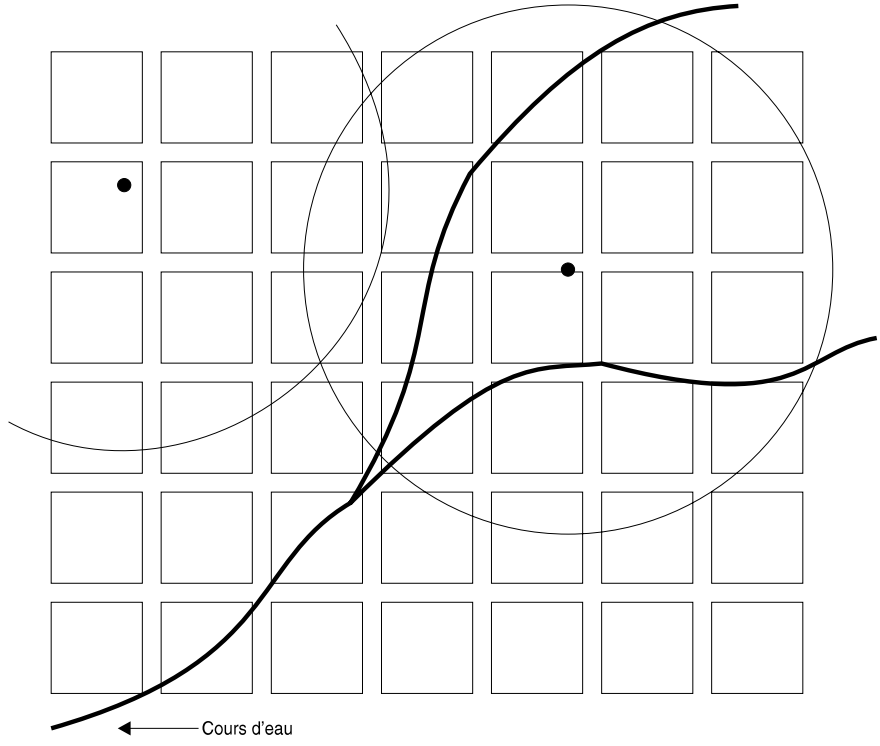


Figure 31. Utilisation du prédicat ST_Crosses pour identifier les cours d'eau qui traversent un zone de déchets dangereux

ST_Difference

ST_Difference utilise deux objets de type géométrie en tant qu'entrée et renvoie un objet de même type résultant de la différence des deux objets source.

Syntaxe

db2gse.ST_Difference(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

db2gse.ST_Geometry

Exemples

Le directeur des services techniques municipaux a besoin de connaître la surface totale non bâtie des parcelles de la ville. Autrement dit, il veut connaître la superficie totale obtenue une fois la surface bâtie soustraite de la surface des parcelles.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                         footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS ( lot_id   integer,
                        lot   db2gse.ST_MultiPolygon);
```

Le directeur des services techniques exécute une équijointure sur les tables BUILDINGFOOTPRINTS et LOTS à partir de l'ID parcelle (lot_id). Il calcule ensuite la surface cumulée issue de la différence parcelles moins bâtis.

```
SELECT
SUM(db2gse.ST_Area(db2gse.ST_Difference(lot,footprint)))
  FROM BUILDINGFOOTPRINTS bf, LOTS
WHERE bf.lot_id = LOTS.lot_id;
```

ST_Dimension

AsShape utilise un objet de type géométrie en tant qu'entrée et renvoie la dimension correspondante.

Syntaxe

db2gse.ST_Dimension(g1 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

La table DIMENSION_TEST est crée avec les colonnes GEOTYPE et G1. La colonne GEOTYPE stocke le nom de la sous-classe de la géométrie enregistrée dans la colonne de géométrie G1.

```
CREATE TABLE DIMENSION_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Les instructions INSERT insèrent une sous-classe exemple dans la table 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)))
```

L'instruction SELECT ci-après affiche le nom de la sous-classe stocké dans la colonne geotype, accompagné de la dimension de ce type de géométrie.

```
SELECT geotype, db2gse.ST_Dimension(g1) "The dimension"
FROM DIMENSION_TEST
```

L'ensemble de résultats suivant est renvoyé :

```
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 utilise deux géométries en tant qu'entrée et renvoie la valeur 1 (TRUE) si l'intersection de deux géométries génère un ensemble vide, et la valeur 0 (FALSE), si tel n'est pas le cas.

Syntaxe

db2gse.ST_Disjoint(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

Une compagnie d'assurances doit estimer le taux de couverture applicable à l'hôpital, aux maisons de retraite et aux écoles d'une ville. Cela consiste en partie à déterminer la menace que les sites de déchets dangereux représentent pour chaque établissement. La compagnie d'assurance ne veut prendre en compte que les établissements ne présentant aucun risque de contamination. Le consultant SIG appointé par la compagnie d'assurances a été chargé de localiser tous les établissements ne figurant pas dans un rayon de 7 km d'un site de produits dangereux.

La table SENSITIVE_AREAS contient plusieurs colonnes qui décrivent les établissements menacés, en sus de la colonne ZONE qui stocke la géométrie de type polygone des établissements.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

La table HAZARDOUS_SITES stocke l'identité des sites dans les colonnes SITE_ID et NAME, alors que l'emplacement géographique réel de chaque site est enregistré dans la colonne LOCATION.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

L'instruction SELECT ci-après répertorie les noms de toutes les zones sensibles qui ne se trouvent pas dans un rayon de 7 km d'un site de déchets dangereux. La fonction ST_Intersects peut se substituer à la fonction ST_Disjoint dans cette requête si le résultat obtenu est égal à 0 au lieu de 1. En effet, ST_Intersects et ST_Disjoint renvoie des résultats exactement inverses.

```
SELECT sa.name
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Disjoint(db2gse.ST_Buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

Sur la figure 32, les zones sensibles sont comparées au rayon de 7 km des sites de déchets dangereux. La maison de retraite est la seule zone sensible pour laquelle la fonction ST_Disjoint renvoie la valeur 1 (TRUE). La fonction ST_Disjoint renvoie cette valeur lorsque deux géométries ne génèrent absolument aucune intersection.

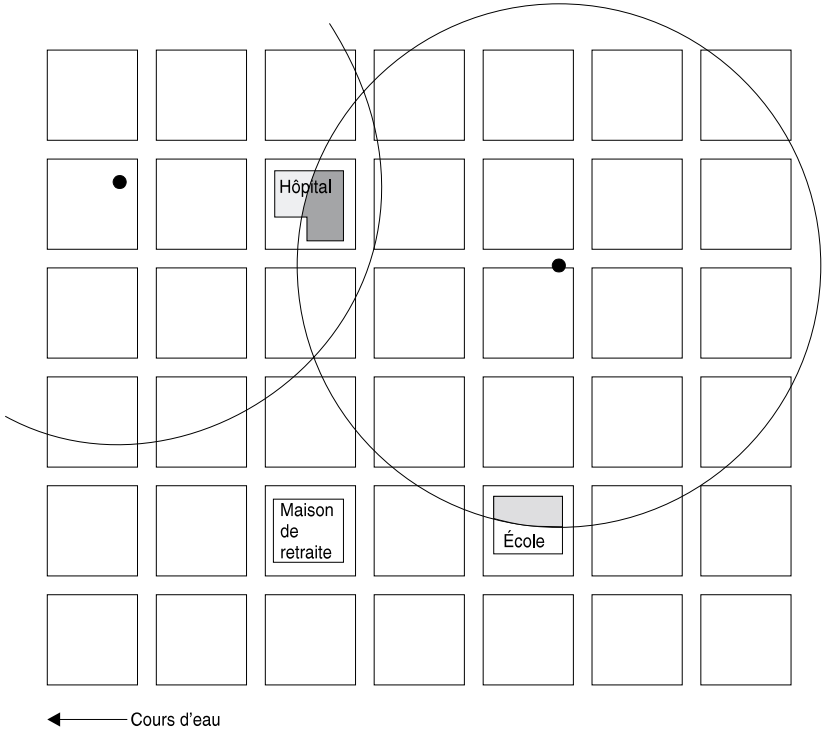


Figure 32. Identification à l'aide de la fonction ST_Disjoint des bâtiments ne figurant à l'intérieur (intersection) d'aucune zone de déchets dangereux

ST_Distance

ST_Distance utilise deux géométries en tant qu'entrée et renvoie la distance minimale les séparant.

Syntaxe

db2gse.ST_Distance(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Double

Exemples

Le directeur des services techniques municipaux a besoin de la liste de tous les bâtiments figurant à mois d'un pied (environ trente centimètres) de toute ligne en pointillé.

La colonne BUILDING_ID de la table BUILDINGFOOTPRINTS identifie chaque bâtiment de manière univoque. La colonne LOT_ID indique la parcelle à laquelle appartient chaque bâtiment. Le multipolygone de bâti enregistre la géométrie de chaque bâti.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                        footprint   db2gse.ST_MultiPolygon);
```

La table LOTS stocke l'ID parcelle (lot_ID) qui identifie chaque parcelle de manière univoque et le multipolygone de la parcelle, qui contient la géométrie des limites de la parcelle.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot      db2gse.ST_MultiPolygon);
```

La requête renvoie la liste des ID des bâtiments situés à moins d'un pied des limites de leur parcelle. La fonction ST_Distance exécute une jointure spatiale entre les bâtis et le contour des multipolygones des parcelles. Cependant, une équijointure entre les colonnes bf.lot_id et LOTS.lot_id permet de s'assurer que la fonction ST_Distance ne compare que les multipolygones appartenant à la même parcelle.

```
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 utilise une ligne en tant qu'entrée et renvoie le dernier point de la ligne.

Syntaxe

db2gse.ST_Endpoint(c db2gse.ST_Curve)

Type de retour

db2gse.ST_Point

Exemples

La table ENDPOINT_TEST contient la colonne de type entier GID qui identifie chaque ligne de manière univoque et la colonne LN1 dans laquelle les lignes sont stockées.

```
CREATE TABLE ENDPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

Les instructions INSERT insèrent des lignes dans la table ENDPOINT_TEST. La première n'a pas de coordonnées Z, ni de mesures alors que la seconde en est dotée.

```
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)))
```

L'instruction SELECT ci-dessous affiche le contenu de la colonne GID accompagné des résultats de la fonction ST_Endpoint. Cette fonction génère une géométrie de type point qui est convertit en texte par la fonction ST_AsText. La fonction CAST permet de raccourcir la valeur varchar(4000) par défaut créée par la fonction ST_AsText en valeur varchar(60).

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.ST_Endpoint(ln1)) AS varchar(60)) "Endpoint"
FROM ENDPOINT_TEST
```

L'ensemble de résultats suivant est renvoyé :

```
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 utilise un objet de type géométrie en tant qu'entrée et renvoie la boîte englobante sous forme de géométrie.

Syntaxe

db2gse.ST_Envelope(g db2gse.ST_Geometry)

Type de retour

db2gse.ST_Geometry

Exemples

La colonne GEOTYPE de la table ENVELOPE_TEST stocke le nom de la sous-classe de la géométrie enregistrée dans la colonne de géométrie G1.

```
CREATE TABLE ENVELOPE_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Les instructions INSERT suivantes insèrent chaque sous-classe de géométrie dans la table 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)))
```

L'instruction SELECT ci-après affiche le nom de la sous-classe à côté de son enveloppe. La fonction ST_Envelope renvoyant un point, une ligne ou un polygone, le résultat généré est converti en texte par la fonction ST_AsText. La fonction CAST permet de convertir la valeur varchar(4000) par défaut créée par la fonction ST_AsText en valeur varchar(280).

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText(db2gse.ST_Envelope(g1))
AS varchar(280)) "The envelope"
FROM ENVELOPE_TEST
```

L'ensemble de résultats suivant est renvoyé :

```
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 compare deux géométries et renvoie la valeur 1 (TRUE) si elles sont identiques, et la valeur 0 (FALSE), dans le cas contraire.

Syntaxe

db2gse.ST_Equals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

Le technicien SIG de la ville pense que certaines données de la table BUILDINGFOOTPRINTS ont été dupliquées d'une manière ou d'une autre. Il analyse la table pour déterminer s'il existe des multipolygones de bâtis égaux.

La table BUILDINGFOOTPRINTS est créée à l'aide de l'instruction ci-après. La colonne BUILDING_ID identifie les bâtiments de manière univoque, la colonne LOT_ID, la parcelle du bâtiment, et la colonne FOOTPRINT contient la géométrie des bâtiments.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                        footprint   db2gse.ST_MultiPolygon);
```

La table BUILDINGFOOTPRINTS est jointe spatialement à elle-même par le prédicat ST_Equals qui renvoie la valeur 1 dès qu'elle trouve deux multipolygones équivalents. La condition bf1.building_id <> bf2.building_id s'impose pour éliminer toute comparaison d'une géométrie avec elle-même.

```
SELECT bf1.building_id, bf2.building_id
FROM BUILDINGFOOTPRINTS bf1, BUILDINGFOOTPRINTS bf2
WHERE db2gse.ST_Equals(bf1.footprint,bf2.footprint) = 1
      and bf1.building_id <> bf2.building_id;
```

ST_ExteriorRing

ST_ExteriorRing utilise un polygone en tant qu'entrée et renvoie son anneau extérieur sous la forme d'une ligne.

Syntaxe

db2gse.ST_ExteriorRing(s db2gse.ST_Polygon)

Type de retour

db2gse.ST_LineString

Exemples

Un ornithologiste, qui étudie le peuplement en oiseaux de plusieurs îles des mers du sud, sait que la zone de nourriture d'une espèce particulière se limite au littoral. Pour calculer le potentiel biotique des îles, il doit calculer leur périmètre. Bien qu'il existe des plans d'eau sur les îles, leurs rives sont exclusivement peuplées par une espèce plus agressive. Par conséquent, l'ornithologiste n'a besoin que du périmètre extérieur des îles.

Les colonnes ID et NAME de la table ISLANDS identifient chaque île et la colonne LAND de type ST_Polygon contient la géométrie de chacune d'elles.

```
CREATE TABLE ISLANDS (id    integer,
                      name  varchar(32),
                      land  db2gse.ST_Polygon);
```

La fonction ST_ExteriorRing extrait l'anneau extérieur de chaque polygone d'île sous forme de ligne. La longueur de la ligne est déterminée par la fonction length. Les longueurs des lignes sont ensuite cumulées à l'aide de la fonction SUM.

```
SELECT SUM(db2gse.ST_length(db2gse.ST_ExteriorRing (land))) FROM ISLANDS;
```

Sur la figure 33 à la page 221, les anneaux extérieurs des îles représentent l'interface écologique que chaque île partage avec la mer. Il existe des lacs sur certaines îles, qui sont représentées par les anneaux intérieurs des polygones.

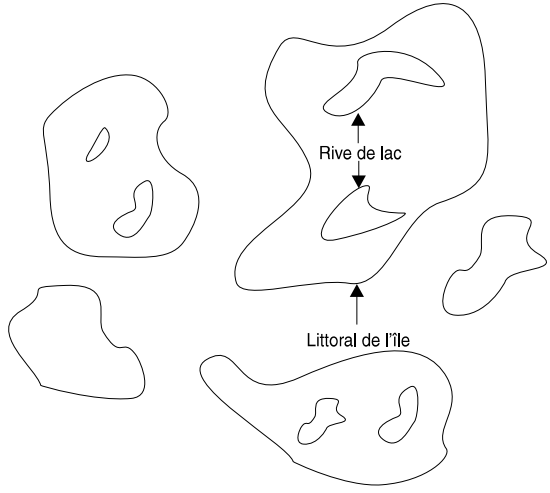


Figure 33. Utilisation de la fonction ST_ExteriorRing pour déterminer la longueur du littoral d'une île

ST_GeometryFromText

ST_GeometryFromText utilise une représentation de texte connue (WKT) et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un objet de type géométrie.

Syntaxe

db2gse.ST_GeometryFromText(geometryTaggedText Varchar(4000), cr db2gse.coordref)

Type de retour

db2gse.ST_Geometry

Exemples

La table GEOMETRY_TEST contient la colonne de type entier GID qui identifie chaque ligne de manière univoque et la colonne G1 qui stocke la géométrie.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Les instructions INSERT insèrent les données dans les colonnes GID et G1 de la table GEOMETRY_TEST. La fonction ST_GeometryFromText convertit la représentation textuelle de chaque géométrie en la sous-classe instanciable DB2 Extension Spatiale correspondante.

```
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 utilise une représentation de type binaire connue (WKB) et un système de références spatiales en tant qu'entrée, et renvoie un objet de type géométrie.

Syntaxe

db2gse.ST_GeomFromWKB(WKBGeometry Blob(1M), cr db2gse.coordref)

Type de retour

db2gse.ST_Geometry

Exemples

Le fragment de code C présenté ci-dessous contient des fonctions ODBC imbriquées dans des fonctions SQL DB2 Extension Spatiale qui insèrent des données dans la table LOTS.

La table LOTS a été créée avec deux colonnes : la colonne LOT_ID qui identifie chaque parcelle de façon univoque et la colonne des multipolygones LOT, qui contient la géométrie de chaque parcelle.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot      db2gse.ST_MultiPolygon);
```

La fonction ST_GeomFromWKB convertit les représentations WKB en géométrie DB2 Extension Spatiale. La totalité de l'instruction INSERT est copiée dans une chaîne de caractères de type wkb_sql. L'instruction INSERT contient des marqueurs de paramètre permettant d'accepter dynamiquement les données LOT_ID et LOT.

```
/* Create the SQL insert statement to populate the lot id and the
   lot multipolygon. The question marks are parameter markers that
   indicate the lot_id and lot values that will be retrieved at
   runtime. */
strcpy (wkb_sql,"insert into LOTS (lot_id, lot) values (?, db2gse.ST_GeomFromWKB

(cast(? as blob(1m)), db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Bind the integer key value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Bind the shape to the second parameter. */
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

ST_GeometryN

ST_GeometryN utilise une collection et un index de type entier en tant qu'entrée, et renvoie le *n*ème objet de type géométrie contenu dans la collection.

Syntaxe

db2gse.ST GeometryN(g db2gse.ST_GeomCollection, n Integer)

Type de retour

db2gse.ST_Geometry

Exemples

Le directeur des services techniques municipaux a besoin de savoir si les bâtis sont tous situés à l'intérieur du premier polygone appartenant au multipolygone de la parcelle.

La colonne BUILDING_ID identifie chaque ligne de la table BUILDINGFOOTPRINTS de manière univoque. La colonne LOT_ID identifie la parcelle du bâtiment et la colonne FOOTPRINT stocke la géométrie de ce dernier.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                        footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS ( lot_id   integer,
                    lot   db2gse.ST_MultiPolygon);
```

La requête répertorie l'ID bâtiment (building_id) et l'ID parcelle (lot_ID) contenus dans la table BUILDINGFOOTPRINTS et associés à tous les bâtis figurant à l'intérieur du premier polygone de la parcelle. La fonction ST_GeometryN renvoie le premier polygone de parcelle appartenant à l'ensemble multipolygonal.

```
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

ST_GeometryType utilise un objet ST_Geometry en tant qu'entrée, dont il renvoie le type sous forme de chaîne.

Syntaxe

db2gse.ST_GeometryType (g db2gse.ST_Geometry)

Type de retour

Varchar(4000)

Exemples

La table GEOMETRYTYPE_TEST contient la colonne de géométrie G1.

```
CREATE TABLE GEOMETRYTYPE_TEST(g1 db2gse.ST_Geometry)
```

Les instructions INSERT suivantes insèrent chaque sous-classe de géométrie dans la colonne 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)))
```

L'instruction SELECT répertorie le type de géométrie de chaque sous-classe enregistrée dans la colonne G1.

```
SELECT db2gse.ST_GeometryType(g1) "Geometry type" FROM GEOMETRYTYPE_TEST
```

L'ensemble de résultats suivant est renvoyé :

```
Geometry type
------------------------------
ST_Point
ST_LineString
ST_Polygon
ST_MultiPoint
ST_MultiLineString
ST_MultiPolygon

6 record(s) selected.
```

ST_InteriorRingN

Renvoie le *n*ème anneau intérieur d'un polygone en tant que ligne. Les anneaux ne sont pas classés en fonction de leur orientation géométrique ; ils le sont selon les règles définies par les routines internes de vérification des géométries. Par conséquent, l'ordre des anneaux ne peut pas être prédéfini.

Syntaxe

ST_InteriorRingN(p ST_Polygon, n Integer)

Type de retour

db2gse.ST_LineString

Exemples

Un ornithologiste, qui étudie le peuplement en oiseaux de plusieurs îles des mers du sud, sait que la zone de nourriture d'une espèce passive particulière se limite au littoral. Il existe plusieurs lacs sur certains îles, dont les rives sont exclusivement peuplées par une autre espèce plus agressive. L'ornithologiste sait que pour chaque île, lorsque le périmètre des lacs dépasse un certain seuil, l'espèce agressive prolifère jusqu'à devenir une menace pour l'espèce côtière passive. Par conséquent, l'ornithologiste a besoin de connaître le périmètre agrégé des anneaux intérieurs (autrement dit des lacs) des îles.

Sur la figure 34, les anneaux extérieurs des îles représentent l'interface écologique que chaque île partage avec la mer. Il existe des lacs sur certaines îles, qui sont représentées par les anneaux intérieurs des polygones.

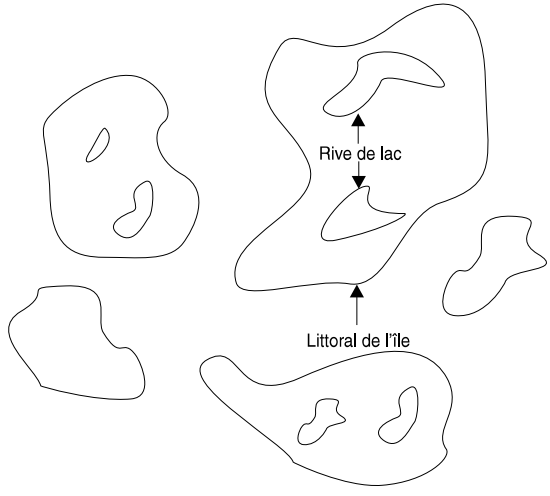


Figure 34. Détermination de la longueur des rives des lacs de chaque île à l'aide de la fonction ST_InteriorRingN

Les colonnes d'ID et de noms de la table ISLANDS identifient chaque île et la colonne des polygones LAND contient la géométrie de l'île.

```
CREATE TABLE ISLANDS (id    integer,
                      name  varchar(32),
                      land  db2gse.ST_Polygon);
```

Le programme ODBC ci-après recourt à la fonction ST_InteriorRingN pour extraire l'anneau intérieur (lac) de chaque polygone d'île en tant que ligne. Le périmètre de la ligne renvoyé par la fonction Length est cumulé et affiché à côté de l'ID de l'île.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include "sg.h"
#include "sgerr.h"
#include "sqlcli1.h"

/***                         ***
 *** Change these constants ***
 ***                         ***/

#define USER_NAME   "sdetest"   /* your user name */
#define USER_PASS   "acid.rain" /* your user password */
#define DB_NAME     "mydb"      /* database to connect to */

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;

/* Allocate memory for the ODBC environment handle henv and
```

```
initialize the application. */

  rc = SQLAllocEnv (&henv);
    if (rc != SQL_SUCCESS)
  {
    printf ("SQLAllocEnv failed with %d\n", rc);
    exit(0);
  }

/* Allocate memory for a connection handle within the henv environment. */

  rc = SQLAllocConnect (henv, &handle);
    if (rc != SQL_SUCCESS)
  {
    printf ("SQLAllocConnect failed with %d\n", rc);
    exit(0);
  }

/* Load the ODBC driver and connect to the data source identified by the database,
   user, and password.*/

  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");

/* Allocate memory to the SQL statement handle island_cursor. */

  rc = SQLAllocStmt (handle, &island_cursor);
  check_sql_err (handle, NULL, rc, "SQLAllocStmt");

/* Prepare and execute the query to get the island IDs and number of
   lakes (interior rings) */

  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");

/* Bind the island table's ID column to the variable island_id */

  rc = SQLBindCol (island_cursor, 1, SQL_C_SLONG, &island_id, 0, &id_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

/* Bind the result of numinteriorrings(land) to the num_lakes variable. */

  rc = SQLBindCol (island_cursor, 2, SQL_C_SLONG, &num_lakes, 0, &lake_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

/* Allocate memory to the SQL statement handle lake_cursor. */
```

```
rc = SQLAllocStmt (handle, &lake_cursor);
  check_sql_err (handle, NULL, rc, "SQLAllocStmt");

/* Prepare the query to get the length of an interior ring. */

  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");

/* Bind the lake_number variable as the first input parameter */

  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");

/* Bind the island_id as the second input parameter */

  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");

/* Bind the result of the
Length(db2gse.ST_InteriorRingN(land, cast
  (? as integer))) to the variable
lake_perimeter */

  rc = SQLBindCol (lake_cursor, 1, SQL_C_SLONG, &lake_perimeter, 0,
                   &length_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

  /* Outer loop, get the island ids and the number of lakes (interior rings) */

  while (SQL_SUCCESS == rc)
  {
    /* Fetch an island */

    rc = SQLFetch (island_cursor);

    if (rc != SQL_NO_DATA)
    {
      check_sql_err (NULL, island_cursor, rc, "SQLFetch");

      /* Inner loop, for this island, get the perimeter of all of
         its lakes (interior rings) */

      for (total_perimeter = 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);
      }
    }

/* Display the Island id and the total perimeter of its lakes. */

    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 utilise un objet de type géométrie en tant qu'entrée et renvoie l'ensemble de l'intersection en tant que géométrie.

Syntaxe

db2gse.ST_Intersection(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

db2gse.ST_Geometry

Exemples

Le commandant des pompiers doit obtenir la liste des zones des hôpitaux, écoles et maisons de retraite qui coupent le rayon d'une zone de contamination potentielle par des déchets dangereux.

Les zones sensibles sont enregistrées dans la table SENSITIVE_AREAS créée avec l'instruction CREATE TABLE ci-dessous. La colonne ZONE est définie en tant que polygone qui contient le contour de chaque zone sensible.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

Les sites à risque sont stockés dans la table HAZARDOUS_SITES créée avec l'instruction CREATE TABLE présentée ci-après. La colonne LOCATION, définie en tant que point, stocke une position représentant le centre géographique de chaque site à risque.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

La fonction Buffer génère une zone tampon de 7 km autour de l'emplacement des sites de déchets dangereux. La fonction ST_Intersection génère des polygones à partir de l'intersection des polygones des zones tampons des sites à risque avec les zones sensibles. La fonction ST_Area renvoie la surface du polygone d'intersection cumulée par la fonction SUM pour chaque site à risque. La clause GROUP BY demande à la requête d'agréger les zones formant une intersection en fonction de l'ID du site de déchets dangereux.

```
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;
```

Sur la figure 35, les cercles représentent les polygones tampons qui entourent les sites de déchets dangereux. L'intersection de ces polygones avec ceux des zones sensibles génèrent trois autres polygones. L'hôpital figurant dans la partie supérieure gauche forme une intersection avec deux cercles alors que que l'école n'en coupe qu'un seul.

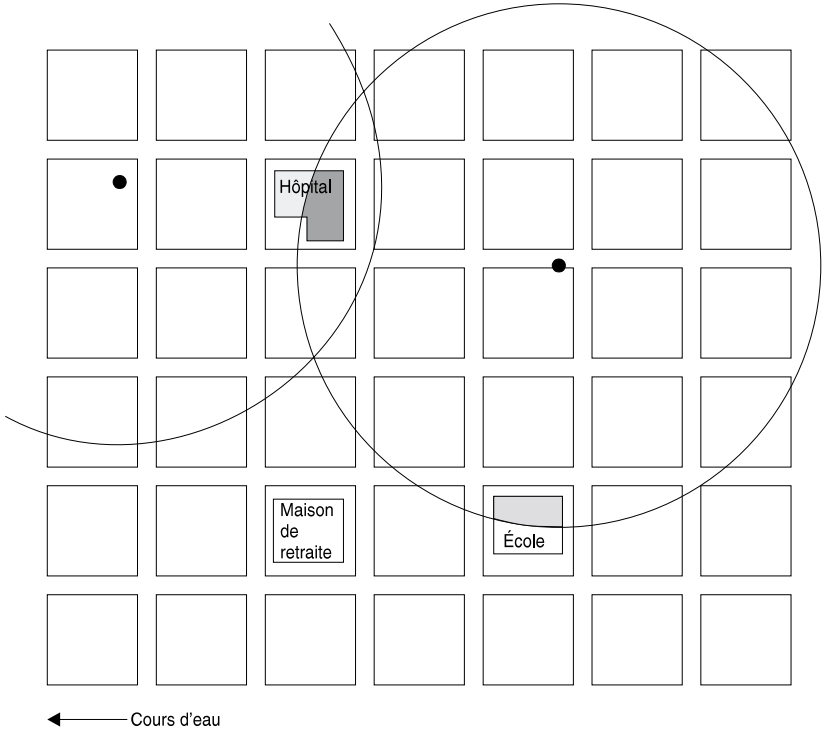


Figure 35. Détermination à l'aide de la fonction ST_Intersection de la surface de chaque bâtiment susceptible d'être affectée par les déchets dangereux

ST_Intersects

ST_Intersects compare deux géométries et renvoie la valeur 1 (TRUE) si l'intersection de deux géométries ne génère pas un ensemble vide, et la valeur 0 (FALSE), dans le cas contraire.

Syntaxe

db2gse.ST_Intersects(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

Le commandant des pompiers a besoin de la liste des zones sensibles situées dans un rayon de 7 km d'un site de déchets dangereux.

Les zones sensibles sont enregistrées dans la table SENSITIVE_AREAS créée avec l'instruction CREATE TABLE ci-dessous. La colonne ZONE est définie en tant que polygone qui contient le contour de chaque zone sensible.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

Les sites à risque sont stockés dans la table HAZARDOUS_SITES créée avec l'instruction CREATE TABLE présentée ci-après. La colonne LOCATION, définie en tant que point, stocke une position représentant le centre géographique de chaque site à risque.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

La requête renvoie la liste des zones sensibles et des noms des sites à risque correspondant qui forment une intersection avec la zone tampon de 7 km définie autour des sites.

```
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 utilise une ligne ou une multiligne en tant qu'entrée et renvoie la valeur 1 (TRUE) si elle est fermée, et la valeur 0 (FALSE), dans le cas contraire.

Syntaxe

db2gse.ST_IsClosed(c db2gse.ST_Curve)
db2gse.ST_IsClosed(mc db2gse.ST_MultiCurve)

Type de retour

Integer

Exemples

L'instruction CREATE TABLE ci-après crée la table CLOSED_LINESTRING, qui comporte une seule colonne de type ligne.

```
CREATE TABLE CLOSED_LINESTRING (ln1 db2gse.ST_LineString)
```

Les instructions INSERT présentées ci-dessous insèrent deux enregistrements dans la table CLOSED_LINESTRING. Le premier n'est pas une ligne fermée alors que le second en est une.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant présentent les résultats de la fonction ST_IsClosed. La première ligne renvoie un 0 car la géométrie de type ligne n'est pas fermée alors que la seconde ligne renvoie un 1 parce que cette géométrie l'est.

```
SELECT db2gse.ST_IsClosed(ln1) "Is it closed" FROM CLOSED_LINESTRING

Is it closed
------------
0
1

  2 record(s) selected.
```

L'instruction CREATE TABLE ci-après crée la table CLOSED_MULTILINESTRING, qui comporte une seule colonne de type multiligne.

```
CREATE TABLE CLOSED_MULTILINESTRING (mln1 db2gse.ST_MultiLineString)
```

Les instructions INSERT présentées ci-dessous insèrent deux enregistrements dans la table CLOSED_MULTILINESTRING : un enregistrement multiligne qui n'est pas fermé et un qui l'est.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant présentent les résultats de la fonction ST_IsClosed. La première ligne renvoie un 0 car la multiligne n'est pas fermée alors que la seconde ligne renvoie un 1 parce que cette multiligne l'est. Une multiligne est fermée si toutes les lignes qui la composent le sont.

```
SELECT db2gse.ST_IsClosed(mln1) "Is it closed" FROM CLOSED_MULTILINESTRING

Is it closed
------------
0
1

  2 record(s) selected.
```

ST_IsEmpty

ST_IsEmpty un objet de type géométrie en tant qu'entrée et renvoie la valeur 1 (TRUE) s'il est vide, et la valeur 0 (FALSE), dans le cas contraire.

Syntaxe

db2gse.ST_IsEmpty(g db2gse.ST_Geometry)

Type de retour

Integer

Exemples

L'instruction CREATE TABLE ci-après crée la table EMPTY_TEST, qui comporte deux colonnes. La colonne GEOTYPE stocke le type de données des sous-classes enregistrées dans la colonne de géométrie G1.

```
CREATE TABLE EMPTY_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Les instructions INSERT présentées ci-dessous insèrent deux enregistrements associés aux sous-classes point, ligne et polygone. Un enregistrement est vide et l'autre ne l'est pas.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant présentent le type de géométrie provenant de la colonne GEOTYPE et les résultats de la fonction 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 utilise une ligne en tant qu'entrée et renvoie la valeur 1 (TRUE) s'il s'agit d'un anneau (autrement dit, si la ligne est fermée et simple), et la valeur 0 (FALSE), dans le cas contraire.

Syntaxe

db2gse.ST_IsRing(c db2gse.ST_Curve)

Type de retour

Integer

Exemples

L'instruction CREATE TABLE ci-après crée la table RING_LINESTRING, qui comporte une seule colonne de type ligne LN1.

```
CREATE TABLE RING_LINESTRING (ln1 db2gse.ST_LineString)
```

Les instructions INSERT ci-après insèrent trois lignes dans la colonne LN1. La première n'est pas fermée et n'est donc pas un anneau, la seconde est fermée et il s'agit donc d'un anneau, et la troisième est fermée mais n'est pas simple puisqu'elle se coupe elle-même, par conséquent, ce n'est pas un anneau.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant présentent les résultats de la fonction ST_IsRing. La première et la troisième lignes renvoient la valeur 0. En effet, ces lignes ne sont pas des anneaux, alors que la deuxième ligne renvoie la valeur 1 car c'en est bien un.

```
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 utilise un objet de type géométrie en tant qu'entrée et renvoie la valeur 1 (TRUE) s'il s'agit d'un objet simple, et la valeur 0 (FALSE), dans le cas contraire.

Syntaxe

db2gse.ST_IsSimple(g db2gse.ST_Geometry)

Type de retour

Integer

Exemples

L'instruction CREATE TABLE ci-après crée la table ISSIMPLE_TEST, qui comporte deux colonnes. La colonne PID de type smallint contient l'identificateur unique de chaque ligne. La colonne de géométrie G1 stocke les géométries exemples simple et non simple.

```
CREATE TABLE ISSIMPLE_TEST (pid smallint, g1 db2gse.ST_Geometry)
```

Les instructions INSERT présentées ci-dessous insèrent deux enregistrements dans la table ISSIMPLE_TEST. Le premier est simple parce qu'il s'agit d'une ligne qui ne se coupe pas elle-même. Le second est complexe par que la ligne génère une intersection avec son propre intérieur.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant présentent les résultats de la fonction ST_IsSimple. Le premier enregistrement renvoie un 1 car la ligne est simple alors que le second renvoie un 0 parce que ce n'est pas le cas de la deuxième ligne.

```
SELECT ST_IsSimple(g1)
FROM ISSIMPLE_TEST

g1
--------------
1
0
```

ST_IsValid

ST_IsValid utilise un objet ST_Geometry en tant qu'entrée et renvoie la valeur 1 (TRUE) s'il est valide, et la valeur 0 (FALSE), dans le cas contraire. Une géométrie insérée dans une base de données DB2 doit toujours être valide car DB2 Extension Spatiale vérifie systématiquement ses données spatiales avant de les accepter. Cependant, il est possible que certains fournisseurs de SGBD ne valident pas leurs entrées mais requièrent que ce soit fait par l'application.

Syntaxe

db2gse.ST_IsValid(g db2gse.ST_Geometry)

Type de retour

Integer

Exemples

La table valid_test est créée avec les colonnes GEOTYPE et G1. La colonne GEOTYPE stocke le nom de la sous-classe de la géométrie enregistrée dans la colonne G1.

```
CREATE TABLE valid_test (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Les instructions INSERT insèrent une sous-classe exemple dans la table 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))
)
```

L'instruction SELECT ci-après répertorie le nom de la sous-classe stocké dans la colonne geotype, accompagné des dimensions de ce type de géométrie.

```
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 utilise une ligne ou une multiligne en tant qu'entrée, et renvoie sa longueur.

Syntaxe

db2gse.ST_Length(c db2gse.ST_Curve)
db2gse.ST_Length(mc db2gse.ST_MultiCurve)

Type de retour

Double

Exemples

Un écologiste local étudie les schémas de migration de la population de saumons dans les cours d'eau du comté. Il veut connaître la longueur de tous les systèmes hydrographiques (rivières et ruisseaux) parcourant le comté.

L'instruction CREATE TABLE ci-après crée la table WATERWAYS_TEST avec les colonnes ID et NAME, qui identifient chaque système hydrographique (rivières et ruisseaux) contenu dans la table. La colonne WATER est de type multiligne car ces systèmes sont souvent un agrégat de plusieurs lignes.

```
CREATE TABLE WATERWAYS (id integer, name
varchar(128),
                            water      db2gse.ST_MultiLineString);
```

L'instruction SELECT ci-après utilise la fonction ST_Length pour renvoyer le nom et la longueur de chaque cours d'eau du comté.

```
SELECT name, db2gse.ST_Length(water) "Length"
FROM WATERWAYS;
```

La figure 36 à la page 248, représente les systèmes de rivières et de ruisseaux existant à l'intérieur des limites du comté.

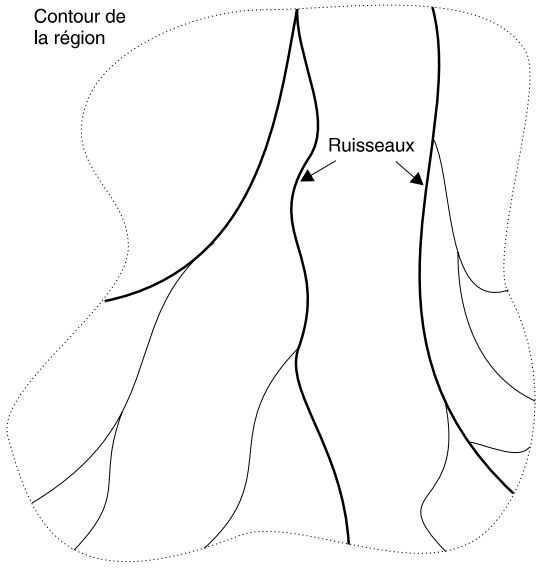


Figure 36. Détermination à l'aide de la fonction ST_Length de la longueur totale des cours d'eau d'un comté

ST_LineFromText

ST_LineFromText utilise une représentation WKT du type ligne et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un objet de type géométrie.

Syntaxe

db2gse.ST_LineFromText(lineStringTaggedText Varchar(4000), cr db2gse.coordref)

Type de retour

db2gse.ST_LineString

Exemples

L'instruction CREATE TABLE ci-après crée la table LINESTRING_TEST, qui comporte une seule colonne de type ligne LN1.

```
CREATE TABLE LINESTRING_TEST (ln1 db2gse.ST_LineString)
```

L'instruction INSERT ci-après insère une géométrie de type ligne dans la colonne LN1 via la fonction 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 utilise une représentation WKB du type ligne et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie une ligne.

Syntaxe

db2gse.ST_LineFromWKB(WKBLineString Blob(1M), cr db2gse.coordref)

Type de retour

db2gse.ST_LineString

Exemples

Le fragment de code ci-après peuple la table SEWERLINES avec l'ID unique, la classe de taille et la géométrie de chaque canalisation d'égout.

La table SEWERLINES_SITINGS est créée avec trois colonnes. La première, SEWER_ID, identifie de manière univoque chaque canalisation. La seconde, CLASS, de type entier identifie le type de canalisation d'égout qui est généralement associé à la capacité de la canalisation. La troisième colonne, SEWER, de type ligne stocke la géométrie de la canalisation d'égout.

```
CREATE TABLE SEWERLINES (sewer_id   integer,
                         class  integer,
                         sewer  db2gse.ST_LineString);

/* Create the SQL insert statement to populate the sewer_id, size class
   and the sewer linestring. The question marks are parameter markers that
   indicate the sewer_id, class and sewer geometry values that will be
   retrieved at runtime. */
strcpy (wkb_sql,"insert into sewerlines (sewer_id,class,sewer)
values (?,?, db2gse.ST_LineFromWKB
(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Bind the integer sewer_id value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_id, 0, &pcbvalue1);

/* Bind the integer class value to the second parameter. */
pcbvalue2 = 0;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_class, 0, &pcbvalue2);

/* Bind the shape to the third parameter. */
pcbvalue3 = blob_len;
```

```
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, sewer_wkb, blob_len, &pcbvalue3);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

ST_MLineFromText

ST_MLineFromText utilise une représentation WKT du type multiligne et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie une multiligne.

Syntaxe

db2gse.ST_MLineFromText(multiLineStringTaggedText String, cr db2gse.coordref)

Type de retour

db2gse.ST_MultiLineString

Exemples

L'instruction CREATE TABLE ci-après crée la table MLINESTRING_TEST qui comporte deux colonnes : la colonne GID de type smallint, qui identifie chaque ligne de la table de manière univoque, et la colonne des multilignes ML1.

```
CREATE TABLE ST_MLINESTRING_TEST (gid smallint, ml1 db2gse.ST_MultiLineString)
```

L'instruction INSERT ci-après insère la multiligne via la fonction 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 utilise une représentation WKB du type multiligne et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie une multiligne.

Syntaxe

db2gse.ST_MLineFromWKB(WKBMultiLineString Blob(1M), cr db2gse.coordref)

Type de retour

db2gse.ST_MultiLineString

Exemples

Le fragment de code ci-après peuple la table WATERWAYS avec un ID unique, un nom et une multiligne.

La table WATERWAYS est créée avec les colonnes ID et NAME, qui identifient chaque système hydrographique (rivières et ruisseaux) contenu dans la table. La colonne WATER est de type multiligne car ces systèmes sont souvent un agrégat de plusieurs lignes.

```
CREATE TABLE WATERWAYS (id        integer,
                        name      varchar(128),
                        water     db2gse.ST_MultiLineString);

/* Create the SQL insert statement to populate the id, name and
   multilinestring. The question marks are parameter markers that
   indicate the id, name and water values that will be retrieved at
   runtime. */
strcpy (shp_sql,"insert into WATERWAYS (id,name,water)
values (?,?, db2gse.ST_MLineFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Bind the integer id value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);

/* Bind the varchar name value to the second parameter. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, name_len, 0, &name, name_len, &pcbvalue2);

/* Bind the shape to the third parameter. */
pcbvalue3 = blob_len;
```

```
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, water_shape, blob_len, &pcbvalue3);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

ST_MPointFromText

ST_MPointFromText utilise une représentation WKT du type multipoint et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un multipoint.

Syntaxe

db2gse.ST_MPointFromText(multiPointTaggedText Varchar(4000), cr db2gse.coordref)

Type de retour

db2gse.ST_MultiPoint

Exemples

L'instruction CREATE TABLE ci-après crée la table MULTIPOINT_TEST qui comporte une seule colonne multipoint, MPT1.

```
CREATE TABLE MULTIPOINT_TEST (mpt1 db2gse.ST_MultiPoint)
```

L'instruction INSERT ci-après insère un multipoint dans la colonne MPT1 via la fonction 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 utilise une représentation WKB du type multipoint et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un multipoint.

Syntaxe

db2gse.ST_MPointFromWKB(WKBMultiPoint Blob(1M), cr db2gse.coordref)

Type de retour

db2gse.ST_MultiPoint

Exemples

Le fragment de code ci-après peuple la table SPECIES_SITINGS.

La table SPECIES_SITINGS est créée avec trois colonnes. Les colonnes SPECIES et GENUS identifient de manière univoque chaque ligne de la table alors que la colonne de multipoints SITINGS représente les lieux d'implantation de l'espèce.

```
CREATE TABLE SPECIES_SITINGS (species  varchar(32),
                              genus varchar(32),
                              sitings  db2gse.ST_MultiPoint);

/* Create the SQL insert statement to populate the species, genus and
   sitings. The question marks are parameter markers that
   indicate the species, genus and sitings values that will be retrieved at
   runtime. */
strcpy (wkb_sql,"insert into SPECIES_SITINGS (species,genus,sitings)
values (?,?, db2gse.ST_MPointFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Bind the varchar species value to the first parameter. */
pcbvalue1 = species_len;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, species_len, 0, &species, species_len, &pcbvalue1);
/* Bind the varchar genus value to the second parameter. */
pcbvalue2 = genus_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, genus_len, 0, &name, genus_len, &pcbvalue2);

/* Bind the shape to the third parameter. */
pcbvalue3 = sitings_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, sitings_len, 0, sitings_wkb, sitings_len, &pcbvalue3);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

ST_MPolyFromText

ST_MPolyFromText utilise une représentation WKT du type multipolygone et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un multipolygone.

Syntaxe

db2gse.ST_MPolyFromText(multiPolygonTaggedText Varchar(4000), cr db2gse.coordref)

Type de retour

db2gse.ST_MultiPolygon

Exemples

L'instruction CREATE TABLE ci-après crée la table MULTIPOLYGON_TEST, qui comporte une seule colonne de type multipolygone, MPL1.

```
CREATE TABLE MULTIPOLYGON_TEST (mpl1 db2gse.ST_MultiPolygon)
```

L'instruction INSERT ci-après insère un multipolygone dans la colonne MPL1 via la fonction 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 utilise une représentation WKB du type multipolygone et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un multipolygone.

Syntaxe

db2gse.ST_MPolyFromWKB(WKBMultiPolygon Blob(1M), cr db2gse.coordref)

Type de retour

db2gse.ST_MultiPolygon

Exemples

Le fragment de code ci-après peuple la table LOTS.

La table LOTS stocke l'ID parcelle (LOT_ID) qui identifie chaque parcelle de manière univoque et le multipolygone de parcelle, qui contient la géométrie de type ligne de la dite parcelle.

```
CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );

/* Create the SQL insert statement to populate the lot_id, and lot. The
   question marks are parameter markers that indicate the lot_id, and lot
   values that will be retrieved at runtime. */
strcpy (wkb_sql,"insert into LOTS (lot_id,lot)
values (?, db2gse.ST_MPolyFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Bind the lot_id integer value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Bind the lot shape to the second parameter. */
pcbvalue2 = lot_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, lot_len, 0, lot_wkb, lot_len, &pcbvalue2);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

ST_NumGeometries

ST_NumGeometries utilise une collection d'objets en entrée et renvoie le nombre de géométries appartenant à la dite collection.

Syntaxe

db2gse.ST_NumGeometries(g db2gse.ST_GeomCollection)

Type de retour

Integer

Exemples

Le directeur des services techniques municipaux a besoin de connaître le nombre de bâtiments distincts associé à chaque bâti.

Les bâtis sont enregistrés dans la table BUILDINGFOOTPRINTS créée avec l'instruction CREATE TABLE ci-après.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

L'instruction SELECT ci-après utilise la fonction ST_NumGeometries pour répertorier l'ID parcelle (BUILDING_ID) qui identifie chaque bâtiment de manière univoque et le nombre de bâtiments contenus dans chaque bâti.

```
SELECT building_id, db2gse.ST_NumGeometries (footprint) "Number of buildings"
FROM BUILDINGFOOTPRINTS;
```

ST_NumInteriorRing

ST_NumInteriorRing utilise un polygone en tant qu'entrée et renvoie le nombre d'anneaux intérieurs que celui-ci contient.

Syntaxe

db2gse.NumInteriorRing(p db2gse.ST_Polygon)

Type de retour

Integer

Exemples

Un ornithologiste, qui étudie le peuplement en oiseaux de plusieurs îles des mers du sud, sait que la zone de nourriture d'une espèce particulière se limite aux îles contenant des lacs d'eau douce. Par conséquent, elle veut connaître le nom des îles qui comportent au moins un lac.

L'instruction CREATE TABLE ci-après crée la table ISLANDS. Les colonnes d'ID et de noms de la table ISLANDS identifient chaque île et la colonne des polygones LAND contient la géométrie de l'île.

```
CREATE TABLE ISLANDS (id integer, name varchar(32), land db2gse.ST_Polygon);
```

Les anneaux intérieurs représentant les lacs, la fonction ST_NumInteriorRing permet restreindre la liste aux îles dotées d'un anneau intérieur, au moins.

```
SELECT name FROM ISLANDS WHERE db2gse.ST_NumInteriorRing(land) > 0;
```

ST_NumPoints

ST_NumPoints utilise une ligne en tant qu'entrée et renvoie le nombre de points la constituant.

Syntaxe

db2gse.ST_NumPoints(l db2gse.ST_LineString)

Type de retour

Integer

Exemples

L'instruction CREATE TABLE ci-après crée la table NUMPOINTS_TEST. La colonne GEOTYPE contient le type de géométrie stocké dans la colonne dans la colonne de géométrie G1.

```
CREATE TABLE NUMPOINTS_TEST (geotype varchar(12), g1 db2gse.ST_Geometry)
```

L'instruction INSERT ci-après insère une ligne.

```
INSERT INTO NUMPOINTS_TEST VALUES( linestring,
db2gse.ST_LineFromText('linestring (10.02 20.01,
23.73 21.92)',
db2gse.coordref()..srid(0)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant présentent les types de géométrie et le nombre de points contenus dans chacun d'eux.

```
SELECT geotype, db2gse.ST_NumPoints(g1)
FROM NUMPOINTS_TEST

GEOTYPE      Number of points
------------ ----------------
ST_linestring          2
  1 record(s) selected.
```

ST_OrderingEquals

ST_OrderingEquals compare deux géométries et renvoie la valeur 1 (TRUE) si elles sont égales et que les coordonnées sont dans le même ordre ; sinon, ce prédicat renvoie la valeur 0 (FALSE).

Syntaxe

db2gse.ST_OrderingEquals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

L'instruction CREATE TABLE ci-après crée la table LINESTRING_TEST, qui comporte deux colonnes de type ligne, LN1 et LN2.

```
CREATE TABLE LINESTRING_TEST (lid integer, l1 db2gse.ST_LineString,
                                                    l2 db2gse.ST_LineString);
```

L'instruction INSERT ci-après insèrent dans les colonnes L1 et L2 deux lignes qui sont égales et dont les coordonnées sont classées dans le même ordre.

```
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)));
```

L'instruction INSERT ci-après insèrent dans les colonnes L1 et L2 deux lignes qui sont égales mais dont les coordonnées ne sont pas classées dans le même ordre.

```
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)));
```

L'instruction SELECT et l'ensemble de résultats ci-après montrent comment la fonction ST_Equals renvoie la valeur 1 (TRUE) sans tenir compte de l'ordre des coordonnées. La fonction ST_OrderingEquals renvoie la valeur 0 (FALSE) lorsque les géométries ne sont pas à la fois égales et dotées de coordonnées classées dans le même ordre.

```
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 utilise deux géométries en tant qu'entrée. Elle renvoie la valeur 1 (TRUE) si l'intersection de ces objets résulte en une géométrie de la même dimension mais non égale à l'un ou l'autre des objets source ; sinon, elle renvoie la valeur 0 (FALSE).

Syntaxe

db2gse.ST_Overlaps(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

L'administrateur du comté a besoin de la liste des sites à risque qui chevauchent des zones sensibles.

L'instruction CREATE TABLE ci-après crée la table SENSITIVE_AREAS. La table SENSITIVE_AREAS contient plusieurs colonnes qui décrivent les établissements menacés, en sus de la colonne ZONE qui stocke la géométrie de type polygone des établissements.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

La table HAZARDOUS_SITES stocke l'identité des sites dans les colonnes SITE_ID et NAME, alors que l'emplacement géographique réel de chaque site est enregistré dans la colonne de type point LOCATION.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

Dans l'instruction SELECT ci-après, les tables SENSITIVE_AREAS et HAZARDOUS_SITES sont jointes à l'aide de la fonction ST_Overlaps. La fonction renvoie la valeur 1 (TRUE) pour toutes les lignes de la table SENSITIVE_AREAS dont les polygones de surface chevauchent le la zone tampon d'un rayon d'environ 7 km défini autour du point représentant l'emplacement HAZARDOUS_SITES.

```
SELECT hs.name
FROM HAZARDOUS_SITES hs, SENSITIVE_AREAS sa
WHERE db2gse.ST_Overlaps (buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

Sur la figure 37 à la page 266, l'hôpital et l'école chevauchent le rayon de deux sites de déchets dangereux du comté, alors que la maison de retraite n'en coupe aucun.

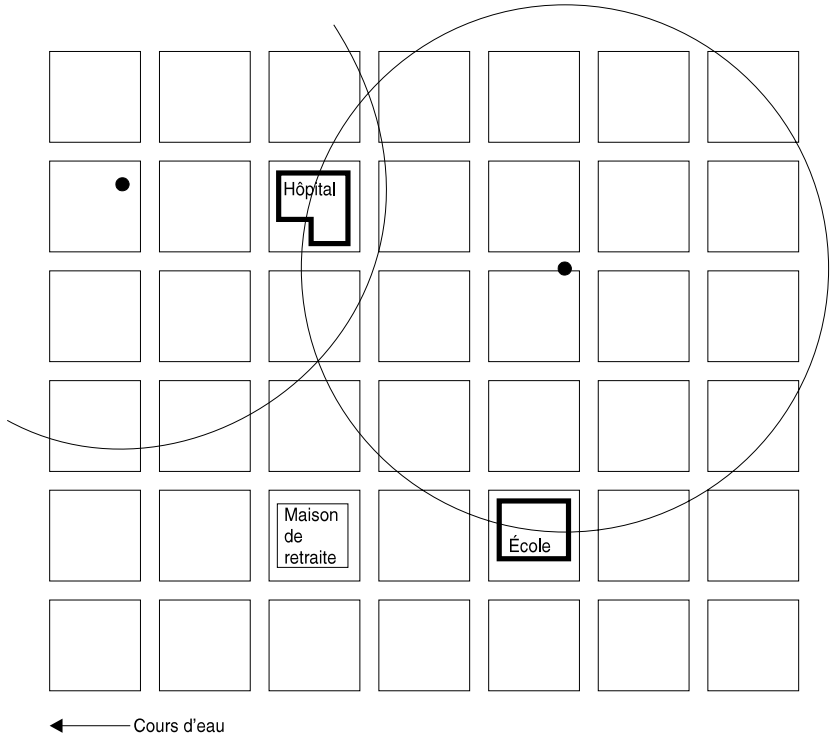


Figure 37. Détermination à l'aide la fonction ST_Overlaps des bâtiments situés partiellement ou totalement à l'intérieur d'une zone de déchets dangereux

ST_Perimeter

ST_Perimeter renvoie le périmètre d'un objet de type ST_Surface.

Syntaxe

db2gse.ST_Perimeter(s db2gse.ST_Surface)
db2gse.ST_Perimeter(ms db2gse.ST_MultiSurface)

Type de retour

Double

Exemples

Un écologiste étudiant les oiseaux vivant sur les rives des plans d'eau doit déterminer la longueur des rives des lacs d'une zone déterminée. Les lacs sont enregistrés en tant que multipolygones dans la table WATERBODIES préalablement créée avec l'instruction CREATE TABLE ci-après.

```
CREATE TABLE WATERBODIES (wbid  integer,
                          waterbody  db2gse.ST_MultiPolygon);
```

Dans l'instruction SELECT ci-après, la fonction ST_Perimeter renvoie le périmètre entourant chaque plan d'eau et la fonction SUM agrège les périmètres avant de renvoyer le total correspondant.

```
SELECT SUM(db2gse.ST_Perimeter(waterbody))
FROM waterbodies;
```

ST_PointFromText

ST_PointFromText utilise une représentation WKT du type point et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un point.

Syntaxe

db2gse.ST_PointFromText(pointTaggedText Varchar(4000), cr db2gse.coordref)

Type de retour

db2gse.ST_Point

Exemples

L'instruction CREATE TABLE ci-après crée la table POINT_TEST, qui comporte une seule colonne de type point, PT1.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

Avant que l'instruction INSERT n'insère le point dans la colonne PT1, la fonction ST_PointFromText convertit les coordonnées de texte du point dans le format point.

```
INSERT INTO POINT_TEST VALUES (
       db2gse.ST_PointFromText ('point(10.01
20.03)',
                                    db2gse.coordref()..srid(0)))
```

ST_PointFromWKB

ST_PointFromWKB utilise une représentation WKB du type point et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un point.

Syntaxe

db2gse.ST_PointFromWKB(WKBPoint Blob(1M), srs SRID)

Type de retour

db2gse.ST_Point

Exemples

Le fragment de code ci-après peuple la table HAZARDOUS_SITES.

Les sites à risque sont stockés dans la table HAZARDOUS_SITES créée avec l'instruction CREATE TABLE présentée ci-après. La colonne LOCATION, définie en tant que point, stocke une position qui représente le centre géographique du site à risque.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);

/* Create the SQL insert statement to populate the site_id, name and
   location. The question marks are parameter markers that indicate the
   site_id, name and location values that will be retrieved at runtime. */
strcpy (wkb_sql,"insert into HAZARDOUS_SITES (site_id, name, location)
values (?,?,
db2gse.ST_PointFromWKB(cast(? as
blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Bind the site_id integer value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);

/* Bind the name varchar value to the second parameter. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Bind the location shape to the third parameter. */
pcbvalue3 = location_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, location_len, 0, location_wkb, location_len, &pcbvalue3);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

ST_Point

ST_Point utilise une abscisse (coordonnée X), une ordonnée (coordonnée Y) et une référence spatiale, et renvoie un objet ST_Point.

Syntaxe

db2gse.ST_Point(X Double, Y Double, srs SRID)

Type de retour

db2gse.ST_Point

Exemples

L'instruction CREATE TABLE ci-après crée la table POINT_TEST, qui comporte une seule colonne de type point, PT1.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

La fonction ST_Point convertit les coordonnées du point en une géométrie de type point avant que l'instruction INSERT ne l'enregistre dans la colonne PT1.

```
INSERT INTO point_test VALUES(
    db2gse.ST_Point(10.01,20.03, db2gse.coordref()..srid(0))
)
```

ST_PointN

ST_PointN utilise une ligne et un index de nombres entiers en tant qu'entrée, et renvoie un point représentant le nème sommet appartenant au tracé de la ligne.

Syntaxe

db2gse.ST_PointN(l db2gse.ST_Curve, n Integer)

Type de retour

db2gse.ST_Point

Exemples

L'instruction CREATE TABLE ci-après crée la table POINTN_TEST qui comporte deux colonnes : la colonne GID qui identifie chaque ligne de la table de manière univoque et la colonne de type ligne LN1.

```
CREATE TABLE POINTN_TEST (gid integer, ln1 db2gse.ST_LineString)
```

Les instructions INSERT ci-après insèrent deux valeurs de ligne. La première n'a pas de coordonnées Z, ni de mesures alors que la seconde est dotée des deux.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant présentent la colonne GID et le second sommet de chaque ligne. La première ligne de la table résulte en un point sans coordonnée Z, ni mesure, alors que le résultat de la seconde est un point doté d'une coordonnée Z et d'une mesure. La fonction ST_PointN renvoie un point avec une coordonnée Z ou une mesure, si ceux-ci existent dans la ligne source.

```
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

ST_PointOnSurface utilise un polygone et un multipolygone en tant qu'entrée, et renvoie un point.

Syntaxe

db2gse.ST_PointOnSurface(s db2gse.ST_Surface)
db2gse.ST_PointOnSurface(ms db2gse.ST_MultiSurface)

Type de retour

db2gse.ST_Point

Exemples

Le directeur des services techniques de la ville doit créer un point doté d'un label pour chaque bâti.

Les bâtis sont enregistrés dans la table BUILDINGFOOTPRINTS créée avec l'instruction CREATE TABLE ci-après.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

La fonction ST_PointOnSurface génère un point qui se trouve avec certitude sur le bâti. Ce point renoyé par la fonction ST_PointOnSurface est ensuite converti par la fonction AsBinaryShape en une forme transtypée en chaîne de caractères de 1 Mo destinée à l'application.

```
SELECT CAST(db2gse.AsBinaryShape(db2gse.ST_PointOnSurface(footprint))
as blob(1m))
FROM BUILDINGFOOTPRINTS;
```

ST_PolyFromText

ST_PolyFromText utilise une représentation WKT du type polygone et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un polygone.

Syntaxe

db2gse.ST_PolyFromText(polygonTaggedText Varchar(4000), cr db2gse.coordref)

Type de retour

db2gse.ST_Polygon

Exemples

L'instruction CREATE TABLE ci-après crée la table POLYGON_TEST qui comporte une seule colonne de type polygone.

```
CREATE TABLE POLYGON_TEST (pl1 db2gse.ST_Polygon)
```

L'instruction INSERT ci-après insère un polygone dans la colonne des polygones via la fonction 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 utilise une représentation WKB du type polygone et l'identité d'un système de références spatiales en tant qu'entrée, et renvoie un polygone.

Syntaxe

db2gse.ST_PolyFromWKB(WKBPolygon Blob(1M), SRID Integer)

Type de retour

db2gse.ST_Polygon

Exemples

Le fragment de code ci-après peuple la table SENSITIVE_AREAS.

Cette table contient plusieurs colonnes qui décrivent les établissements menacés, en sus de la colonne zone qui stocke la géométrie de type polygone des établissements.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);


/* Create the SQL insert statement to populate the id, name, size, type and
   zone. The question marks are parameter markers that indicate the id,name,
   size, type and zone values that will be retrieved at runtime. */
strcpy (shp_wkb,"insert into SENSITIVE_AREAS (id, name, size, type, zone)
values (?,?,?,?,
db2gse.ST_PolyFromWKB (cast(? as
blob(1m)),
db2gse.coordref()..srid(0)))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */
rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Bind the id integer value to the first parameter. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);

/* Bind the name varchar value to the second parameter. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Bind the size float to the third parameter. */
pcbvalue3 = 0;
```

```
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_FLOAT,
     SQL_REAL, 0, 0, &size, 0, &pcbvalue3);

/* Bind the type varchar to the fourth parameter. */
pcbvalue4 = type_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_VARCHAR, type_len, 0, type, type_len, &pcbvalue4);

/* Bind the zone polygon to the fifth parameter. */
pcbvalue5 = zone_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, zone_len, 0, zone_wkb, zone_len, &pcbvalue5);

/* Execute the insert statement. */
rc = SQLExecute (hstmt);
```

ST_Polygon

ST_Polygon génère un objet de type ST_Polygon à partir d'un objet de type ST_LineString et d'un ID de système de références spatiales.

Syntaxe

db2gse.ST_Polygon(l db2gse.ST_LineString, cr db2gse.coordref)

Type de retour

db2gse.ST_Polygon

Exemples

L'instruction CREATE TABLE ci-après crée la table POLYGON_TEST qui comporte une seule colonne, P1.

```
CREATE TABLE POLYGON_TEST (p1 db2gse.ST_polygon)
```

L'instruction INSERT ci-après convertit un anneau (ligne fermée et simple) en un polygone et l'insère dans la colonne P1 à l'aide de la fonction ST_LineFromText imbriquée dans la fonction 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 compare deux géométries et renvoie la valeur 1 (TRUE) si elles remplissent les conditions spécifiées par la chaîne de la matrice de schémas DE-9IM ; sinon, ce prédicat renvoie la valeur 0 (FALSE). Pour plus d'informations sur les matrices du modèle DE-9IM, reportez-vous à la section «Prédicats» à la page 143.

Syntaxe

db2gse.ST_Relate(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry, patternMatrix String)

Type de retour

Integer

Exemples

Une matrice de schémas DE-9IM est un outil de comparaison des géométries. Il existe plusieurs types de matrices de ce genre. Par exemple, la matrice d'*égalité* vous indique si deux géométries quelconques sont égales.

En l'occurrence, une matrice de schémas d'égalité, illustrée au tableau 56, se lit de gauche à droite et de haut en bas sous forme de chaîne («T*F**FFF*»).

Tableau 56. Matrice des schémas d'égalité

		b		
		Intérieur	**Contour**	**Extérieur**
	Intérieur	T	*	F
a	**Contour**	*	*	F
	Extérieur	F	F	*

Ensuite, la table RELATE_TEST est créée avec l'instruction CREATE TABLE ci-après.

```
CREATE TABLE RELATE_TEST (rid integer, g1 db2gse.ST_Geometry,
g2 db2gse.ST_Geometry, g3 db2gse.ST_Geometry);
```

Les instructions INSERT insèrent une sous-classe exemple dans la table 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)
)
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant affichent le nom de la sous-classe stocké dans la colonne geotype, accompagné de la dimension de ce type de géométrie.

```
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 utilise un objet de type géométrie en tant qu'entrée et renvoie l'identité de son système de références spatiales.

Syntaxe

db2gse.ST_SRID(g1 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

La table SPATIAL_REFERENCES est créée au cours de l'installation de DB2 Extension Spatiale. Lors de la création d'une géométrie, l'ID de la référence spatiale (SRID) associée est entré dans la table SPATIAL_REFERENCES. La fonction ST_SRID renvoie la valeur de cette entrée.

Par exemple, un type de géométrie est utilisé dans l'instruction CREATE TABLE :

```
CREATE TABLE SRID_TEST(g1 db2gse.ST_Geometry)
```

Dans l'instruction INSERT ci-après, une géométrie de type point localisée à la coordonnée 10.01,50.76 est insérée dans la colonne des géométrie G1. Lorsque le point a été créé par la fonction ST_PointFromText, la valeur de SRID 1 lui est affectée.

```
INSERT INTO SRID_TEST
VALUES
(db2gse.ST_PointFromText('point(10.01
50.76)',
                                    db2gse.coordref()..srid(0)))
```

La fonction ST_SRID renvoie l'identité du système de références spatiales associé à la géométrie qui vient d'être enregistrée, comme illustré par l'instruction SELECT ci-après et l'ensemble de résultats correspondant.

```
SELECT db2gse.ST_SRID(g1) FROM SRID_TEST

g1
--------------
1
```

ST_StartPoint

ST_StartPoint utilise une ligne en tant qu'entrée et renvoie le premier point de la ligne.

Syntaxe

db2gse.ST_StartPoint(c db2gse.ST_Curve)

Type de retour

db2gse.ST_Point

Exemples

L'instruction CREATE TABLE ci-après crée la table STARTPOINT_TEST. Cette table comporte deux colonnes : la colonne de type entier GID qui identifie chaque ligne de la table de manière univoque et la colonne de type ligne LN1.

```
CREATE TABLE STARTPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

Les instructions INSERT ci-après insèrent les lignes dans la colonne LN1. La première n'a pas de coordonnées Z, ni de mesures alors que la seconde est dotée des deux.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant montrent comment la fonction ST_StartPoint extrait le premier point de chaque ligne. Cette fonction convertit le point dans son format texte. Le premier point de la liste n'a ni coordonnée Z, ni mesure, alors que le second est doté des deux parce que sa ligne source en avait également.

```
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

ST_SymmetricDiff utilise deux objets de type géométrie en tant qu'entrée et renvoie un objet de même type représentant la différence symétrique des deux objets source.

Syntaxe

db2gse.ST_SymmetricDiff(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

db2gse.ST_Geometry

Exemples

L'administrateur du comté doit déterminer la surface des zones sensibles et du rayon de 7 km entourant les sites à risque qui ne forment pas d'intersection.

L'instruction CREATE TABLE ci-après crée la table SENSITIVE_AREAS, qui comporte plusieurs colonnes décrivant les établissements menacés. Cette table contient également la colonne ZONE qui stocke la géométrie de type polygone des établissements.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

L'instruction CREATE TABLE ci-après crée la table HAZARDOUS_SITES qui stocke l'identité des sites dans les colonnes SITE_ID et NAME. Quant à l'emplacement géographique réel de chaque site, il est enregistré dans la colonne de type point LOCATION.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  point);
```

La fonction ST_Buffer génère une zone tampon de 7 km autour de l'emplacement des sites de déchets dangereux. La fonction ST_SymmetricDiff génère des polygones à partir de l'intersection des polygones des zones tampons des sites à risque avec les zones sensibles. La fonction ST_Area renvoie la surface du polygone d'intersection pour chaque site à risque.

```
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
```

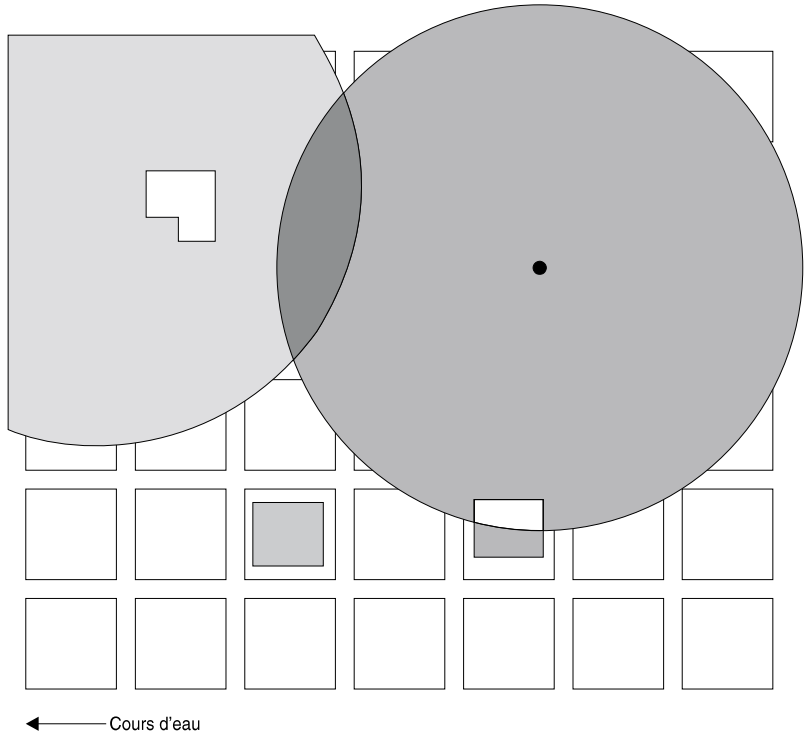


Figure 38. Détermination à l'aide de la fonction ST_SymmetricDiff des zones de déchets dangereux ne contenant pas de zones sensibles (bâtiments habités)

Sur la figure 38, la différence symétrique des sites de déchets dangereux et des zones sensibles est obtenue en soustrayant les zones d'intersection de la surface totale.

ST_Touches

ST_Touches renvoie la valeur 1 (TRUE) si aucun des points communs aux deux géométries ne forme d'intersection avec les intérieurs des deux géométries. Sinon, il renvoie la valeur 0 (FALSE). Une des deux géométries, au moins, doit être une ligne, un polygone, une multiligne ou un multipolygone.

Syntaxe

db2gse.ST_Touches(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

Le technicien SIG doit fournir la liste de toutes les canalisations d'égout dont les extrémités forment une intersection avec une autre canalisation.

L'instruction CREATE TABLE présentée ci-après crée la table SEWERLINES qui comporte trois colonnes. La première, SEWER_ID, identifie de manière univoque chaque canalisation. La seconde, CLASS, de type entier identifie le type de canalisation d'égout qui est généralement associé à la capacité de la canalisation. La troisième colonne, SEWER, de type ligne stocke la géométrie de la canalisation d'égout.

```
CREATE TABLE SEWERLINES (sewer_id integer, class integer,
sewer db2gse.ST_LineString);
```

L'instruction SELECT ci-après renvoie une liste ordonnée des ID canalisation (SEWER_ID) dont les extrémités se touchent.

```
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

ST_Transform affecte une géométrie à un système de références spatiales différent de celui auquel elle est actuellement affectée.

Syntaxe

db2gse.ST_Transform(g db2gse.ST_Geometry, cr db2gse.coordref)

Type de retour

db2gse.ST_Geometry

Exemples

L'instruction CREATE TABLE ci-après crée la table TRANSFORM_TEST, qui comporte deux colonnes de type ligne, LN1 et LN2.

```
CREATE TABLE TRANSFORM_TEST (tid integer, l1 db2gse.ST_LineString,
l2 db2gse.ST_LineString)
```

L'instruction INSERT ci-après insère une ligne dans la colonne l1 avec un 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)
```

La fonction ST_Transform convertit la ligne contenue dans la colonne L1 de la référence de coordonnée affectée au SRID 102 en la référence de coordonnée affectée au SRID 105. L'instruction UPDATE ci-après enregistre la ligne transformée dans la colonne l2.

```
UPDATE TRANSFORM_TEST SET l2 = db2gse.ST_Transform(l1,
db2gse.coordref()..srid(105))
```

ST_Union

ST_Union utilise deux objets de type géométrie en tant qu'entrée et renvoie un objet de même type résultant de l'union des deux objets source.

Syntaxe

db2gse.ST_Union(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

db2gse.ST_Geometry

Exemples

L'instruction CREATE TABLE ci-après crée la table SENSITIVE_AREAS, qui comporte plusieurs colonnes décrivant les établissements menacés. Cette table contient également la colonne ZONE qui stocke la géométrie de type polygone des établissements.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

L'instruction CREATE TABLE ci-après crée la table HAZARDOUS_SITES, qui stocke l'identité des sites dans les colonnes SITE_ID et NAME. Quant à l'emplacement géographique réel de chaque site, il est enregistré dans la colonne de type point LOCATION.

```
CREATE TABLE HAZARDOUS_SITES (site_id integer, name varchar(128),
                              location  db2gse.ST_Point);
```

L'instruction SELECT ci-après utilise la fonction ST_Buffer pour créer une zone tampon de 7 km autour de l'emplacement des sites de déchets dangereux. La fonction ST_Union génère des polygones à partir de l'union des polygones des zones tampons des sites à risque et les zones sensibles. La fonction ST_Area renvoie l'union de la surface des polygones.

```
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 utilise deux géométries en tant qu'entrée et renvoie la valeur 1 (TRUE) si le premier objet est entièrement contenu entièrement dans le second et la valeur 0 (FALSE), si tel n'est pas le cas.

Syntaxe

db2gse.ST_Within(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Type de retour

Integer

Exemples

Dans l'exemple ci-dessous, deux tables sont créées. La première, BUILDINGFOOTPRINTS, contient le bâti d'une ville et la seconde, LOTS, le parcellaire. Le directeur des services techniques municipaux veut s'assurer que tous les bâtis sont entièrement contenus dans leur parcelle respective.

Dans les deux tables, le type de données multipolygone permet de stocker la géométrie des bâtis et celle des parcelles. Le concepteur de la base de données a tenu compte du fait que les parcelles pouvaient être disjointes par des entités naturelles (rivière, etc.) et qu'un bâti pouvait souvent être constitué de plusieurs bâtiments.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);


CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

L'instruction SELECT ci-après a permis au directeur des services techniques municipaux de sélectionner tout d'abord les bâtis qui n'étaient pas entièrement contenus dans une parcelle.

```
SELECT building_id
  FROM BUILDINGFOOTPRINTS, LOTS
 WHERE db2gse.ST_Within(footprint,lot) = 0;
```

Bien que la première requête renvoie la liste de tous les ID bâtiment associés à des bâtis figurant en dehors d'un polygone de parcelle, elle ne permet pas de déterminer si les autres se sont vus affecter des ID parcelle (lot_id) corrects. La seconde instruction SELECT exécute une vérification de l'intégrité des données sur la colonne LOT_ID de la table 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 crée une valeur ST_Geometry à partir de sa représentation WKB. La valeur SRID 0 est automatiquement utilisée.

Syntaxe

db2gse.ST_WKBToSQL(WKBGeometry Blob(1M))

Type de retour

db2gse.ST_Geometry

Exemples

L'instruction CREATE TABLE ci-après crée la table LOTS avec les deux colonnes suivantes : la colonne LOT_ID qui identifie chaque parcelle de façon univoque et la colonne des multipolygones LOT, qui contient la géométrie de chaque parcelle.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

Le fragment de code C présenté ci-dessous contient des fonctions ODBC imbriquées dans des fonctions SQL DB2 Extension Spatiale qui insèrent des données dans la table LOTS.

La fonction ST_WKBToSQL convertit les représentations WKB en géométrie DB2 Extension Spatiale. La totalité de l'instruction INSERT est copiée dans une chaîne de caractères de type wkb_sql. L'instruction INSERT contient des marqueurs de paramètre permettant d'accepter dynamiquement les données LOT_ID et LOT.

```
/* Create the SQL insert statement to populate the lot id and the
   lot multipolygon. The question marks are parameter markers that
   indicate the lot_id and lot values that will be retrieved at
   run time. */

strcpy (wkb_sql,"insert into lots (lot_id, lot)
  values(?, db2gse.ST_WKBToSQL(cast(? as blob(1m))))");

/* Allocate memory for the SQL statement handle and associate the
   statement handle with the connection handle. */

rc = SQLAllocStmt (handle, &hstmt);

/* Prepare the SQL statement for execution. */

rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Bind the integer key value to the first parameter. */

pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);
```

```
/* Bind the shape to the second parameter. */

pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Execute the insert statement. */

rc = SQLExecute (hstmt);
```

ST_WKTToSQL

ST_WKTToSQL crée une valeur ST_Geometry à partir de sa représentation WKT. La valeur SRID 0 est automatiquement utilisée.

Syntaxe

db2gse.ST_WKTToSQL(geometryTaggedText Varchar(4000))

Type de retour

db2gse.ST_Geometry

Exemples

L'instruction CREATE TABLE ci-après crée la table GEOMETRY_TEST qui comporte deux colonnes : la colonne GID de type entier qui identifie chaque ligne de la table de manière univoque et la colonne de type G1 qui stocke la géométrie.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Les instructions INSERT insèrent les données dans les colonnes GID et G1 de la table GEOMETRY_TEST. La fonction ST_WKTToSQL convertit la représentation textuelle de chaque géométrie en la sous-classe instanciable DB2 Extension Spatiale correspondante.

```
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 utilise un point en tant qu'entrée et renvoie une abscisse (X).

Syntaxe

ST_X(p ST_Point)

Type de retour

Double

Exemples

L'instruction CREATE TABLE ci-après crée la table X_TEST qui comporte deux colonnes : la colonne GID qui identifie chaque ligne de la table de manière univoque et la colonne de type point PT1.

```
CREATE TABLE X_TEST (gid integer, pt1 db2gse.ST_Point)
```

Les instructions INSERT ci-après insèrent deux lignes de table. La première est un point sans coordonnée Z, ni mesure, alors que la seconde est un point doté des deux.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant affichent les données de la colonne GID et l'abscisse à double précision des points.

```
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 utilise un point en tant qu'entrée et renvoie une ordonnée (Y).

Syntaxe

db2gse.ST_Y(p db2gse.ST_Point)

Type de retour

Double

Exemples

L'instruction CREATE TABLE ci-après crée la table Y_TEST qui comporte deux colonnes : la colonne GID qui identifie chaque ligne de la table de manière univoque et la colonne de type point PT1.

```
CREATE TABLE Y_TEST (gid integer, pt1 db2gse.ST_Point)
```

Les instructions INSERT ci-après insèrent deux lignes de table. La première est un point sans coordonnée Z, ni mesure, alors que la seconde est un point doté des deux.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant affichent les données de la colonne GID et l'ordonnée à double précision des points.

```
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 utilise un point en tant qu'entrée et renvoie une coordonnée Z.

Syntaxe

Z(p db2gse.ST_Point)

Type de retour

Double

Exemples

L'instruction CREATE TABLE ci-après crée la table Z_TEST qui comporte deux colonnes : la colonne GID qui identifie chaque ligne de la table de manière univoque et la colonne de type point PT1.

```
CREATE TABLE Z_TEST (gid integer, pt1 db2gse.ST_Point)
```

Les instructions INSERT ci-après insèrent deux lignes de table. La première est un point sans coordonnée Z, ni mesure, alors que la seconde est un point doté des deux.

```
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)))
```

L'instruction SELECT ci-après et l'ensemble de résultats correspondant affichent les données de la colonne GID et la coordonnée Z à double précision des points. La première ligne n'est associée à aucune valeur car le point n'a pas de coordonnée 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.
```

Chapitre 15. Systèmes de coordonnées

Le présent chapitre fournit des informations de référence sur le système de références spatiales (SRS) et les valeurs de coordonnées utilisées pour l'interprétation de données spatiales.

Présentation des systèmes de coordonnées

La représentation textuelle connue (WKT - well-known text) des systèmes de références spatiales fournit une représentation standard de type texte des informations que ceux-ci contiennent. Les définitions de cette représentation sont créées sur le modèle de données de systèmes de coordonnées POSC/EPSG.

Un système de références spatiales est un système de coordonnées géographiques (latitude-longitude), projetées (X,Y) ou géocentriques (X,Y,Z). Le système de coordonnées est composé de plusieurs objets. Chacun d'eux est associé à un mot clé en majuscules (par exemple, DATUM ou UNIT) suivi des paramètres de définition de l'objet délimités par des virgules et figurant entre parenthèses. Certains objets sont eux-mêmes composés de plusieurs objets et on obtient donc une structure imbriquée.

Remarque : Dans les implémentations, les parenthèses normales ( ) peuvent remplacer les crochets [ ] et ces deux typographies doivent donc pouvoir être lues.

La définition EBNF (Extended Backus Naur Form) de la réprésentation de type chaîne d'un système de coordonnées utilisant des crochets est la suivante (reportez-vous à la remarque précédente sur l'utilisation des crochets) :

```
<système de coordonnées> = <sc projetées>
| <sc géographiques> | <sc géocentriques>
<sc projetées> =
PROJCS["<nom>", <sc géographiques>,
<projection>, {<paramètre>,}*
               <unité linéaire>]
<projection> = PROJECTION["<nom>"]
<paramètre> = PARAMETER["<nom>", <valeur>]
<valeur> = <nombre>
```

Le système de coordonnées d'un ensemble de données est connu par le mot clé PROJCS si les données sont dans des coordonnées projetées (par GEOGCS et GOCCS si elles sont respectivement en coordonnées géographiques ou en coordonnées géocentriques). Le mot clé PROJCS est suivi par tous les éléments (pieces) qui définissent le système de coordonnées projetées. Le premier élément d'un objet est toujours le nom. Plusieurs objets suivent le nom du système de coordonnées projetées : le système de coordonnées géographiques, la projection cartographique, un voire plusieurs paramètres et l'unité de mesure linéaire. Tous les systèmes de coordonnées projetées sont dérivés d'un système de coordonnées géographiques. Par conséquent, la présente section décrit tout d'abord les éléments spécifiques d'un système de coordonnées projetées. Par exemple, la zone UTM (Universal Transverse Mercator) fuseau 10 nord sur le datum NAD83 est ainsi définie :

```
PROJCS["NAD_1983_UTM_Zone_10N",
<sc géographiques>,
PROJECTION["Transverse_Mercator"],
PARAMETER["False_Easting",500000.0],
PARAMETER["False_Northing",0.0],
PARAMETER["Central_Meridian",-123.0],
PARAMETER["Scale_Factor",0.9996],
PARAMETER["Latitude_of_Origin",0.0],
UNIT["Meter",1.0]]
```

Le nom associé à plusieurs objets définissent l'objet système de coordonnées géographiques : la référence, le méridien origine et l'unité angulaire.

```
<sc géographiques> = GEOGCS["<nom>", <datum>, <méridient origine>, <unité angulaire>]
<datum> = DATUM["<nom>", <sphéroïde>]
<sphéroïde> = SPHEROID["<nom>", <demi grand axe>, <aplatissement inverse>]
<demi grand axe> = <nombre>
    (Le demi grand axe est mesuré en mètres et doit être > 0.)
<aplatissement inverse> = <nombre>
<méridien origine> = PRIMEM["<nom>", <longitude>]
<longitude> = <nombre>
```

Chaîne du système de coordonnées géographique pour la zone UTM fuseau 10 sur NAD83 :

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],
UNIT["Degree",0.0174532925199433]]
```

L'objet UNIT peut représenter des unités de mesure linéaire ou angulaire :

```
<unité angulaire> = <unité>
<unité linéaire> = <unité>
<unité> = UNIT["<nom>", <facteur de conversion>]
<facteur de conversion> = <nombre>
```

Le facteur de conversion spécifie le nombre de mètres (pour une unité linéaire) ou le nombre de radians (pour une unité d'angle) par unité et il doit être supérieur à zéro.

Par conséquent la totalité de la représentation de type chaîne de la zone UTM 10N est la suivante :

```
PROJCS["NAD_1983_UTM_Zone_10N",
GEOGCS["GCS_North_American_1983",
DATUM[ "D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433]],
PROJECTION["Transverse_Mercator"],PARAMETER["False_Easting",500000.0],
PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",-123.0],
PARAMETER["Scale_Factor",0.9996],PARAMETER["Latitude_of_Origin",0.0],
UNIT["Meter",1.0]]
```

Un système de coordonnées géocentriques ressemble beaucoup à un système de coordonnées géographiques :

```
<sc géocentriques> = GEOCCS["<nom>", <datum>, <méridien origine>,
<unité linéaire>]
```

Unités linéaires prises en charge

Tableau 57. Unités linéaires prises en charge

Unité	Facteur de conversion
Mètre	1,0
Pied (Foot) (International)	0,3048
Pied (Etats-Unis)	12/39,37
Pied américain modifié	12,0004584/39,37
Pied de Clarke	12/39,370432

Tableau 57. Unités linéaires prises en charge (suite)

Unité	Facteur de conversion
Pied indien	12/39,370141
Link	7,92/39,370432
Link (Benoit)	7,92/39,370113
Link (Sears)	7,92/39,370147
Chain (Benoit)	792/39,370113
Chain (Sears)	792/39,370147
Yard (Indien)	36/39,370141
Yard (Sears)	36/39,370147
Brasse	1,8288
Mille marin	1852,0

Unités angulaires prises en charge

Tableau 58. Unités angulaires prises en charge

Unité	Facteur de conversion
Radian	1,0
Degré décimal	p/180
Minute décimale	(p/180)/60
Seconde décimale	(p/180)/36000
Gon	p/200
Grade	p/200

Spéroïdes pris en charge

Tableau 59. Sphéroïdes pris en charge

Nom	Demi-grand axe	Inverses de l'aplatissement
Airy	6377563,396	299,3249646
Modified Airy	6377340,189	299,3249646
Australian	6378160	298,25
Bessel	6377397,155	299,1528128
Modified Bessel	6377492,018	299,1528128
Bessel (Namibie)	6377483,865	299,1528128

Tableau 59. Sphéroïdes pris en charge (suite)

Nom	Demi-grand axe	Inverses de l'aplatissement
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
Everest 1830	6377276,345	300,8017
Everest 1975	6377301,243	300,8017
Everest (Sarawak et Sabah)	6377298,556	300,8017
Modified Everest 1948	6377304,063	300,8017
Fischer 1960	6378166	298,3
Fischer 1968	6378150	298,3
Fischer rectifié (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
International 1909	6378388	297,0
International 1924	6378388	297,0
New International 1967	6378157,5	298,2496
Krasovsky	6378245	298,3
Mercury 1960	6378166	298,3
Modified Mercury 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
South American 1969	6378160	298,25
Southeast Asia	6378155	298,3

Tableau 59. Sphéroïdes pris en charge (suite)

Nom	Demi-grand axe	Inverses de l'aplatissement
Sphère (rayon = 1,0)	1	0
Sphère (rayon = 6371000 m)	6371000	0
Sphère (rayon =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
WGS 1972	6378135	298,26
WGS 1984	6378137	298,257223563

Référentiels géodésiques pris en charge

Tableau 60. Références géodésiques prises en charge

Adindan	Lisbon
Afgooye	Loma Quintana
Agadez	Lomé
Australian Geodetic Datum 1966	Luzon 1911
Australian Geodetic Datum 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

Tableau 60. Références géodésiques prises en charge (suite)

Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Bogota	Nord de Guerre
Bukit Rimpah	NGO 1948
Camacupa	Nord Sahara 1959
Campo Inchauspe	NSWC 9Z-2
Cape	Nouvelle Triangulation Francaise
Carthage	New Zealand Geodetic Datum 1949
Chua	OS (SN) 1980
Conakry 1905	OSGB 1936
Corrego Alegre	OSGB 1970 (SN)
Cote d'Ivoire	Padang 1884
Datum 73	Palestine 1923
Deir ez Zor	Pointe Noire
Deutsche Hauptdreiecksnetz	Provisional South American Datum 1956
Douala	Pulkovo 1942
European Datum 1950	Qatar
European Datum 1987	Qatar 1948
Egypt 1907	Qornoq
European Reference System 1989	RT38
Fahud	South American Datum 1969
Gandajika 1970	Sapper Hill 1943
Garoua	Schwarzeck
Geocentric Datum of Australia 1994	Segora
Guyane Francaise	Serindung
Herat North	Stockholm 1938
Hito XVIII 1963	Sudan
Hu Tzu	Shan Tananarive 1925
Hungarian Datum 1972	Timbalai 1948
Indian 1954	TM65
Indian 1975	TM75
Indonesian Datum 1974	Tokyo
Jamaica 1875	Trinidad 1903

Tableau 60. Références géodésiques prises en charge (suite)

Jamaica 1969	Trucial Coast 1948
Kalianpur	Voirol 1875
Kandawala	Voirol Unifie 1960
Kertau	WGS 1972
Kuwait Oil Company	WGS 1972 Transit Broadcast Ephemeris
La Canoa	WGS 1984
Lake	Yacare
Leigon	Yoff
Liberia 1964	Zanderij

Méridiens origine pris en charge

Tableau 61. Méridiens origine pris en charge

Lieu	Coordonnées
Greenwich	0° 0' 0"
Bern	7° 26' 22,5" E
Bogota	74° 4' 51,3" O
Bruxelles	4° 22' 4,71" E
Ferro	17° 40' 0" O
Djakarta	106° 48' 27,79" E
Lisbonne	9° 7' 54,862" O
Madrid	3° 41' 16,58" O
Paris	2° 20' 14,025" E
Rome	12° 27' 8,4" E
Stockholm	18° 3' 29" E

Projections cartographiques carte prises en charge

Tableau 62. Projections cartographiques prises en charge

Projections cylindriques	Projections pseudo-cylindriques
Behrmann	Parabolique de Craster
Cassini	Eckert I
Cylindrique équivalente	Eckert II

Tableau 62. Projections cartographiques prises en charge (suite)

Projections cylindriques	Projections pseudo-cylindriques
Equiréctangulaire	Eckert III
Stéréographique de Gall	Eckert IV
Gauss-Kruger	Eckert V
Mercator	Eckert VI
Cylindrique de Miller	Quadratique polaire plane de McBryde-Thomas
Oblique	Mercator (Hotine) Mollweide
Plate-Carée	Robinson
Times	Sinusoïdale (Sansom-Flamsteed)
Transverse de Mercator	Winkel I

Projections coniques

Tableau 63. Projections coniques

Conique équivalente d'Albers	Trimétrique de Chamberlin
Conique conforme oblique bipolaire	Equidistante en deux points
Bonne	Equivalente de Hammer-Aitoff
Conique équidistante	Van der Grinten I
Conique conforme de Lambert	Diverses
Polyconique	Alaska série E
Conique simple	Grille Alaska (stéréographique rectifiée par Snyder)

Projections azimutales ou planes

- Azimutale équidistante
- Perspective générale verticale
- Gnomonique
- Azimutale équivalente de Lambert
- Orthographique
- Stéréographique-polaire
- Stéréographique

Paramètres de projection cartographique

Tableau 64. Paramètres de projection cartographique

Paramètre	Description
central_meridian	Longitude choisie en tant qu'origine des coordonnées X.
scale_factor	Généralement utilisé pour réduire le degré de distorsion dans une projection cartographique.
standard_parallel_1	Latitude généralement sans distorsion. Egalement utilisé pour le paramètre "latitude d'échelle réelle".
standard_parallel_2	Latitude généralement sans distorsion.
longitude_of_center	Longitude qui définit le point central de la projection cartographique.
latitude_of_center	Latitude qui définit le point central de la projection cartographique.
latitude_of_origin	Latitude choisie en tant qu'origine des ordonnées (Y).
false_easting	Constante ajoutée aux abscisses. Utilisée pour obtenir des valeurs positives.
false_northing	Constante ajoutée aux ordonnées. Utilisée pour obtenir des valeurs positives.
azimuth	Angle à l'est du nord qui définit la ligne centrale d'une projection oblique.
longitude_of_point_1	Longitude du premier point requis pour une projection cartographique.
latitude_of_point_1	Latitude du premier point requis pour une projection cartographique.
longitude_of_point_2	Longitude du second point requis pour une projection cartographique.
latitude_of_point_2	Latitude du second point requis pour une projection cartographique.
longitude_of_point_3	Longitude du troisième point requis pour une projection cartographique.
latitude_of_point_3	Latitude du troisième point requis pour une projection cartographique.
landsat_number	Numéro d'un satellite Landsat.
path_number	Numéro de la trajectoire orbitale d'un satellite déterminé.

Tableau 64. Paramètres de projection cartographique (suite)

Paramètre	Description
perspective_point_height	Hauteur au dessus de la terre d'un point de perspective d'une projection cartographique.
fipszone	Numéro de zone provenant du système de coordonnées State Plane??
zone	Numéro de zone UTM.

Chapitre 16. Formats de fichiers pour données spatiales

Le présent chapitre décrit les représentations connues DB2 Extension Spatiale. Elles sont appelées ainsi (*well-known*) parce qu'elles sont fournies par l'ESRI et ne sont pas spécifiques de DB2 Extension Spatiale. Il existe trois sortes de valeurs spatiales qu'il est important de comprendre pour les processus d'importation et d'exportation de données spatiales.

- Les représentations textuelles connues (WKT - well-known text) du groupement Open GIS Consortium (OGC)
- Les représentations binaires connues (WKB - well-known binary) de l'OGC
- Les représentations de forme ESRI

Représentations textuelles connues (WK)T de l'OGC

DB2 Extension Spatiale dispose de plusieurs fonctions de génération de géométries à partir de descriptions textuelles :

ST_GeomFromText
Crée une géométrie à partir de la représentation textuelle d'un type de géométrie quelconque.

ST_PointFromText
Crée un point à partir de la représentation textuelle d'un point.

ST_LineFromText
Crée une ligne à partir de la représentation textuelle d'une ligne.

ST_PolyFromText
Crée un polygone à partir de la représentation textuelle d'un polygone.

ST_MPointFromText
Crée un multipoint à partir de la représentation textuelle d'un multipoint.

ST_MLineFromText
Crée un multiligne à partir de la représentation textuelle d'une multiligne.

ST_MPolyFromText
Crée un multipolygone à partir de la représentation textuelle d'un multipolygone.

La représentation textuelle consiste en une chaîne de format de texte ASCII qui permet d'échanger une géométrie en format de texte ASCII. Vous pouvez

utiliser ces fonctions dans un programme de troisième ou quatrième génération (3GL ou 4GL) parce qu'elles n'impliquent pas de définir des structures de programmes spéciales. La fonction ST_AsText convertit une valeur de géométrie existante en une représentation textuelle.

Chaque type de géométrie dispose de sa propre représentation textuelle connue (WKT) qui peut être utilisée pour créer de nouvelles instances de ce type et pour convertir des instances existantes en un format de type texte en vue d'un affichage alphanumérique.

La représentation textuelle connue d'une géométrie est définie comme suit : la notation {}* signale zéro ou plus répétitions des marques à l'intérieur des parenthèses. les parenthèses ne figurent pas dans la liste des marques de la sortie.

```
<Geometry Tagged Text> :=
| <Point Tagged Text>
| <LineString Tagged Text>
| <Polygon Tagged Text>
| <MultiPoint Tagged  Text>
| <MultiLineString Tagged Text>
| <MultiPolygon Tagged Text>

<Point Tagged Text> :=
POINT <Point Text>

<LineString Tagged Text> :=
LINESTRING <LineString Text>

<Polygon Tagged Text> :=
POLYGON <Polygon Text>

<MultiPoint Tagged Text> :=
MULTIPOINT <Multipoint Text>

<MultiLineString Tagged Text> :=
MULTILINESTRING <MultiLineString Text>

<MultiPolygon Tagged Text> :=
MULTIPOLYGON <MultiPolygon Text>

<Point Text> := EMPTY
|    <Point>
| Z  <PointZ>
| M  <PointM>
| ZM <PointZM>

<Point> :=  <x>  <y>
<x> := double precision literal
<y> := double precision literal
<PointZ> :=  <x>  <y>  <z>
<x> := double precision literal
<y> := double precision literal
```

```
<z> := double precision literal
<PointM> :=  <x>  <y>  <m>
<x> := double precision literal
<y> := double precision literal
<m> := double precision literal
<PointZM> :=  <x>  <y>  <z>  <m>
<x> := double precision literal
<y> := double precision literal
<z> := double precision literal
<m> := double precision literal

<LineString Text> := EMPTY
|    ( <Point Text >   {,  <Point Text> }*  )
| Z  ( <PointZ Text >  {,  <PointZ Text> }*  )
| M  ( <PointM Text >  {,  <PointM Text> }*  )
| ZM ( <PointZM Text > {,  <PointZM Text> }*  )

<Polygon Text> := EMPTY
| ( <LineString Text > {,< LineString Text > }*  )

<Multipoint Text> := EMPTY
| ( <Point Text >   {,  <Point Text > }*  )

<MultiLineString Text> := EMPTY
| ( <LineString Text > {,< LineString Text>}*  )

<MultiPolygon Text> := EMPTY
| ( < Polygon Text > {,  < Polygon Text > }*  )
```

La syntaxe de base de la fonction est la suivante :

```
function (<text description>,<SRID>)
```

Le SRID, identificateur du système de références spatiales et clé primaire pour la table SPATIAL_REFERENCES, identifie le système de références spatiales de la géométrie stocké dans la table SPATIAL_REFERENCES. Avant qu'une géométrie soit insérée dans une colonne spatiale, son SRID doit correspondre à celui associé à la colonne spatiale.

La description textuelle est constituée de trois éléments de base figurant entre apostrophes par exemple :

```
<'type-géométrie'>
['type-coordonnées']
['liste-coordonnées']
```

où :

type-géométrie
: Est l'un des types suivants : point, ligne, polygone, multipoint, multiligne ou multipolygone.

type coordonnées
: Spécifie si la géométrie est dotée de coordonnées Z ou de mesures.

Laissez cet argument vide si ce n'est pas le cas. Sinon, définissez le type de coordonnées par Z pour les géométries contenant des coordonnées Z, par M pour celles dotées de mesures et par ZM pour les géométries dotées des deux.

liste-coordonnées
Définit les sommets de la géométrie. Les listes de coordonnées sont délimitées par des virgules et comprises entre parenthèses. Les géométries composées d'éléments multiples requièrent l'utilisation de parenthèses pour englober la partie consacrée à chaque élément. Si la géométrie est vide, le mot clé EMPTY remplace la coordonnée.

Le tableau 65 présente une liste exhaustive de toutes les représentations textuelles possibles.

Tableau 65. Types de géométrie et représentations textuelles associées

Type de géométrie	Description textuelle	Commentaire
point	point empty	point vide
point	point z empty	point vide avec coordonnée Z
point	point m empty	point vide avec mesure
point	point zm empty	point vide avec coordonnée Z et mesure
point	point (10.05 10.28)	point
point	point z (10.05 10.28 2.51)	point avec coordonnée Z
point	point m (10.05 10.28 4.72)	point avec mesure
point	point zm (10.05 10.28 2.51 4.72)	point avec coordonnée Z et mesure
ligne	linestring empty	ligne vide
ligne	linestring z empty	ligne vide avec coordonnées Z
ligne	linestring m empty	ligne vide avec mesures
ligne	linestring zm empty	ligne vide avec coordonnées Z et mesures
ligne	linestring (10.05 10.28 , 20.95 20.89)	ligne
ligne	linestring z (10.05 10.28 3.09, 20.95 31.98 4.72, 21.98 29.80 3.51)	ligne avec coordonnées Z
ligne	linestring m (10.05 10.28 5.84, 20.95 31.98 9.01, 21.98 29.80 12.84)	ligne avec mesures

Tableau 65. Types de géométrie et représentations textuelles associées (suite)

Type de géométrie	Description textuelle	Commentaire
ligne	linestring zm ( )	ligne avec coordonnées Z et mesures
polygone	polygon empty	polygone vide
polygone	polygon z empty	polygone vide avec coordonnées z
polygone	polygon m empty	polygone vide avec mesures
polygone	polygon zm empty	polygone vide avec coordonnées Z et mesures
polygone	polygon ((10 10, 10 20, 20 20, 20 15, 10 10))	polygone
polygone	polygon z (( ))	polygone avec coordonnées z
polygone	polygon m (( ))	polygone avec mesures
polygone	polygon zm (( ))	polygone avec coordonnées Z et mesures
multipoint	multipoint empty	multipoint vide
multipoint	multipoint z empty	multipoint vide avec coordonnées z
multipoint	multipoint m empty	multipoint vide avec mesures
multipoint	multipoint zm empty	multipoint vide avec coordonnées z et mesures
multipoint	multipoint empty	multipoint vide
multipoint	multipoint (10 10, 20 20)	multipoint à deux points
multipoint	multipoint z (10 10 2, 20 20 3)	multipoint avec coordonnées z
multipoint	multipoint m (10 10 4, 20 20 5)	multipoint avec mesures
multipoint	multipoint zm (10 10 2 4, 20 20 3 5)	multipoint avec coordonnées Z et mesures
multiligne	multilinestring empty	multiligne vide
multiligne	multilinestring z empty	multiligne vide avec coordonnées z
multiligne	multilinestring m empty	multiligne vide avec mesures

Tableau 65. Types de géométrie et représentations textuelles associées (suite)

Type de géométrie	Description textuelle	Commentaire
multiligne	multilinestring zm empty	multiligne vide avec coordonnées z et mesures
multiligne	multilinestring (( ))	multiligne
multiligne	multilinestring z (( ))	multiligne avec coordonnées z
multiligne	multilinestring m (( ))	multiligne avec mesures
multiligne	multilinestring zm (( ))	multiligne avec coordonnées z et mesures
multipolygone	multipolygon empty	multipolygone vide
multipolygone	multipolygon z empty	multipolygone vide avec coordonnées z
multipolygone	multipolygon m empty	multipolygone vide avec mesures
multipolygone	multipolygon z	multipolygone vide avec coordonnées z et mesures
multipolygone	multipolygon ((( )))	multipolygone
multipolygone	multipolygon z ((( )))	multipolygone avec coordonnées z
multipolygone	multipolygon 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)))	multipolygone avec mesures
multipolygone	multipolygon zm ((( )))	multipolygone avec coordonnées z et mesures

Représentations binaires connues (WKB - well-known binary) de l'OGC

DB2 Extension Spatiale dispose de plusieurs fonctions de génération de géométries à partir de représentations binaires :

ST_GeomFromWKB
Crée une géométrie à partir d'une représentation binaire connue (WKB) d'un type de géométrie quelconque.

ST_PointFromWKB
Crée un point à partir de la représentation binaire d'un point.

ST_LineFromWKB
Crée une ligne à partir de la représentation binaire d'une ligne.

ST_PolyFromWKB
Crée un polygone à partir de la représentation textuelle d'un polygone.

ST_MPointFromWKB
Crée un multipoint à partir de la représentation binaire d'un multipoint.

ST_MLineFromWKB
Crée une multiligne à partir de la représentation binaire d'une multiligne.

ST_MPolyFromWKB
Crée un multipolygone à partir de la représentation binaire d'un multipolygone.

La représentation binaire connue est un flot contigu d'octets. Elle permet d'échanger une géométrie entre un client ODBC et une base de données SQL dans un format binaire. Ces fonctions impliquent de définir des structures C pour le mappage de la représentation binaire ; elles sont donc destinées à être utilisées dans un programme de langage de troisième génération (3GL). Elles ne conviennent pas à un environnement de langage de quatrième génération (4GL). La fonction ST_AsBinary convertit une valeur de géométrie existante en une représentation binaire connue (WKB).

La représentation binaire connue d'une géométrie est obtenue en sérialisant une instance de la géométrie en tant que séquence de types numériques. Ces types sont extraits de l'ensemble (unsigned integer, double) et chacun d'eux est ensuite sérialisé en tant que séquence d'octets. Les types sont sérialisés en utilisant une des deux normes de représentation binaire connue associées aux types numériques (NDR et XDR). Une balise d'un octet précédant les bits sérialisés décrit le codage binaire spécifique (NDR ou XDR) appliqué au train d'octets d'une géométrie. Ces deux systèmes de codage des géométries ne se distinguent que par l'ordre des octets : le codage XDR est de type big-endian alors que le codage NDR est de type little-endian.

Définitions des types numériques

Un *entier non signé (unsigned integer)* est un type de données sur 32 bits (4 octets), qui encode un entier non négatif appartenant à la plage [0, 4294967295].

Un *double* est un type de données à double précision sur 64 bits (8 octets), qui encode un nombre à double précision en recourant au format de double précision IEEE 754.

Ces définitions s'appliquent aux deux formats, XDR et NDR.

Types numériques associés au codage XDR (Big Endian)

La représentation XDR d'un entier non signé est de type big-endian (octet le plus significatif en premier).

La représentation XDR d'un double est de type big-endian (le bit de signe constitue le premier octet).

Types numériques associés au codage NDR (Little Endian)

La représentation NDR d'un entier non signé est de type little-endian (octet le moins significatif en premier).

La représentation NDR d'un double est de type little-endian (le bit de signe constitue le dernier octet).

Conversion entre NDR et XDR

La conversion entre les types de données NDR et XDR exécutée sur les entiers non signés et les doubles est une opération simple. Elle consiste à inverser l'ordre des octets au sein de chaque entier non signé ou double appartenant au train d'octets.

Description des trains d'octets WKBGeometry

La présente section décrit la représentation binaire connue (WKB) associée aux géométries. Le bloc de construction de base est le train d'octet d'un point, qui est composé de deux doubles. Les trains d'octets des autres géométries sont créés à partir des trains d'octets de géométries déjà définies.

```
// Basic Type definitions
// byte : 1 byte
// uint32 : 32 bit unsigned integer  (4 bytes)
// double : double precision number (8 bytes)

// Building Blocks : Point, LinearRing

Point {
  double x;
  double y;
};
LinearRing   {
  uint32  numPoints;
  Point   points[numPoints];
};
enum wkbGeometryType {
  wkbPoint = 1,
  wkbLineString = 2,
  wkbPolygon = 3,
  wkbMultiPoint = 4,
  wkbMultiLineString = 5,
  wkbMultiPolygon = 6,
};
enum wkbByteOrder {
  wkbXDR = 0,                                    // Big Endian
```

```
  wkbNDR = 1                                    // Little Endian
};
WKBPoint {
  byte     byteOrder;
  uint32   wkbType;                                          // 1
  Point    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;
  }
};
```

La figure ci-après illustre une représentation NDR.

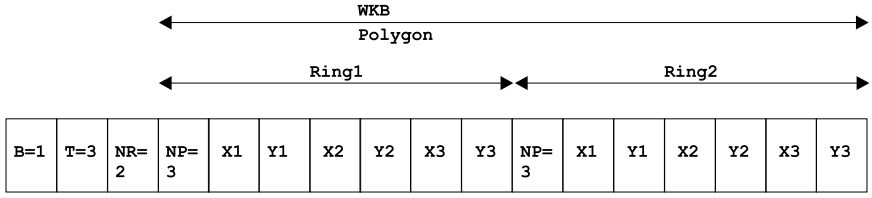


Figure 39. Représentation au format NDR. (B=1) de type polygone (T=3) avec 2 anneaux linéaires (NR=2), chaque anneau ayant 3 points (NP=3).

Assertions concernant la représentation WKB

La représentation binaire connue (WKB) des géométries est conçue pour représenter des instances des types de géométrie décrits dans le modèle d'objet géométrie (Geometry Object Model) et dans la Spécification abstraite OpenGIS.

Ces assertions ont les répercutions suivantes sur les anneaux, polygones et multipolygones :

Anneaux linéaires
Les anneaux sont simples et fermés, autrement dit, des anneaux linéaires ne peuvent pas former d'intersection avec eux-mêmes.

Polygones
Deux anneaux linéaires appartenant au contour d'un polygone ne doivent pas se couper. Les anneaux linéaires du contour d'un polygone peuvent au plus former une intersection en un seul point mais uniquement si celui-ci est tangent.

Multipolygones
Les intérieurs de deux polygones composant un multipolygone ne peuvent pas former d'intersection. Les contours de deux polygones appartenant à un multipolygone ne peuvent être en contact qu'en un nombre fini de points.

Représentation de forme ESRI

DB2 Extension Spatiale dispose de plusieurs fonctions de génération de géométries à partir de représentations de formes ESRI. La représentation de formes ESRI prend en charge les coordonnées Z et les mesures, en sus de la représentation bidimensionnelle prise en charge par la représentation binaires connue OpenGis. Les fonctions suivantes génèrent une géométrie à partir d'une forme ESRI :

ST_GeometryFromShape
Crée une géométrie à partir de représentation en tant que forme d'un type de géométrie quelconque.

ST_PointFromShape
Crée un point à partir de la représentation de type forme d'un point.

ST_LineFromShape
Crée une ligne à partir de la représentation de type forme d'une ligne.

ST_PolyFromShape
Crée un polygone à partir de la représentation de type forme d'un polygone.

ST_MPointFromShape
Crée un multipoint à partir de la représentation de type forme d'un multipoint.

ST_MLineFromShape
Crée un multiligne à partir de la représentation de type forme d'une multiligne.

ST_MPolyFromShape
Crée un multipolygone à partir de la représentation de type forme d'un multipolygone.

La syntaxe générale de ces fonctions est identique. Le premier argument est la représentation de la forme entrée en tant que type de données BLOB. Le second argument est l'entier identifiant la référence spatiale à affecter à la géométrie. La fonction GeometryFromShape doit respecter la syntaxe suivante :

```
db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)
```

Ces fonctions SHAPE impliquent de définir des structures C pour le mappage de la représentation binaire ; elles sont donc destinées à être utilisées dans un programme de langage de troisième génération (3GL). La fonction AsShape convertit la valeur d'une géométrie en une représentation de forme ESRI.

Un type de forme de valeur 0 indique une forme vide (NULL) sans données géométriques associées à la forme.

Valeur	Shape Type
0	Null Shape
1	Point
3*	PolyLine
5	Polygon
8	MultiPoint
11	PointZ
13	PolyLineZ

Valeur	Shape Type
15	PolygonZ
18	MultiPointZ
21	PointM
23	PolyLineM
25	PolygonM
28	MultiPointM

Remarque : *Les types de forme qui ne sont pas spécifiés ci-dessus (2, 4, 6, etc.) sont réservés à de futures utilisations.

Types de formes dans un espace XY

Point

Un point consiste en une paire de coordonnées à double précision dans l'ordre X, Y.

Tableau 66. Contenu du train d'octets d'un point

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	1	Integer	1	Little
Octet 4	X	X	Double	1	Little
Octet 12	Y	Y	Double	1	Little

MultiPoint

Un multipoint consiste en une collection de points. La boîte englobante est stockée dans l'ordre suivant : Ymin, Ymin, Xmax, Ymax.

Tableau 67. Contenu du train d'octets d'un multipoint

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	8	Integer	1	Little
Octet 4	Box	Box	Double	4	Little
Octet 36	NumPoints	NumPoints	Integer	1	Little
Octet 40	Points	Points	Point	NumPoints	Little

PolyLine

Une polyligne est un ensemble ordonné de sommets, composé d'une ou de plusieurs parties. Une partie est une suite connectée de plusieurs points (deux ou plus). Les points peuvent être ou non connectés les uns aux autres. Des parties peuvent ou non se croiser.

Cette spécification n'interdit pas les points consécutifs dotés de coordonnées identiques, les lecteurs des fichiers SHAPE doivent donc traiter ces cas. Cependant, les parties dégénérées de longueur zéro susceptibles d'être ainsi produites ne sont pas autorisées.

Les zones d'une polyligne sont les suivantes :

Box Boîte englobante associée à la polyligne et stockée dans l'ordre suivant : Ymin, Ymin, Xmax, Ymax.

NumParts
Nombre de parties de la polyligne.

NumPoints
Nombre total de points pour toutes les parties.

Parts Tableau de longueur NumParts. Chaque polyligne stocke l'indice de son premier point dans le tableau des points. Les indices de tableau sont indiqués par rapport à 0.

Points Tableau de longueur NumPoints. Les points de chaque partie de la polyligne sont stockés d'extrémité en extrémité. Les points de la deuxième partie suivent ceux de la première, etc. Le tableau des parties contient l'indice de tableau du point de départ de chaque partie. Il n'existe pas de délimiteur entre parties dans le tableau des points.

Tableau 68. Contenu du train d'octets d'une polyligne

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	3	Integer	1	Little
Octet 4	Box	Box	Double	4	Little
Octet 36	NumParts	NumParts	Integer	1	Little
Octet 40	NumPoints	NumPoints	Integer	1	Little
Octet 44	Parts	Parts	Integer	NumParts	Little
Octet X	Points	Points	Point	NumPoints	Little

Remarque : X = 44 + 4 * NumParts.

Polygon

Un polygone consiste en un ou plusieurs anneaux. Un anneau est une séquence connectée de quatre points ou plus, qui forment une boucle fermée qui ne génère pas d'intersection avec elle-même. Un polygone peut contenir plusieurs anneaux externes. L'ordre des sommets ou orientation d'un anneau indique de quel côté de l'anneau se situe l'intérieur du polygone. La zone de voisinage située à droite d'un observateur marchant le long de l'anneau dans l'ordre des sommets représente l'intérieur du polygone. Les sommets des

anneaux qui définissent des trous dans les polygones vont dans une direction contraire au sens des aiguilles d'une montre. Les sommets d'un polygone à anneaux doivent être considérées dans le sens des aiguilles d'une montre. Les anneaux d'un polygone sont appelées parties.

Cette spécification n'interdit pas les points consécutifs dotés de coordonnées identiques, les lecteurs des fichiers SHAPE doivent donc traiter ces cas. Cependant, les parties dégénérées de longueur zéro ou de surface zéro susceptibles d'être ainsi produites ne sont pas autorisées.

Les zones d'un polygone sont les suivantes :

Box Boîte englobante associée au polygone et stockée dans l'ordre suivant : Ymin, Ymin, Xmax, Ymax.

NumParts
Nombre d'anneaux compris dans le polygone.

NumPoints
Nombre total de points pour tous les anneaux.

Parts Tableau de longueur NumParts. Stocke, pour chaque anneau, l'indice de son premier point dans le tableau des points. Les indices de tableau sont indiqués par rapport à 0.

Points Tableau de longueur NumPoints. Les points de chaque anneau du polyligone sont stockés d'extrémité en extrémité. Les points du deuxième anneau suivent ceux du premier, etc. Le tableau des parties contient l'indice de tableau du point de départ de chaque anneau. Il n'existe pas de délimiteur entre anneaux dans le tableau des points.

Remarques importantes sur les formes de type polygone :

- Les anneaux sont fermés (le premier sommet d'un anneau est aussi le dernier).
- L'ordre des anneaux dans le tableau des points n'a pas d'importance.
- Les polygones stockés dans un fichier Shape doivent être nettoyés. Un polygone est nettoyé quand :
  - il ne forme aucune insersection avec lui-même. Cela signifie qu'un segment appartenant à l'un des anneaux ne peut pas former d'intersection avec un segment d'un autre anneau. Les anneaux d'un polygone peuvent être en contact à leurs sommets mais pas le long de segments. Les segments alignés sont considérés comme formant une intersection.
  - l'intérieur du dit polygone se trouve du côté ″correct″ de la ligne qui le définit. La zone de voisinage située à droite d'un observateur marchant le long de l'anneau en suivant dans l'ordre des sommets représente l'intérieur du polygone. Les sommets

d'un polygone seul à anneaux doivent être considérées dans le sens des aiguilles d'une montre. Les anneaux définissant des trous dans ces polygones sont dotés d'une orientation contraire aux sens des aiguilles d'une montre.

Des polygones ″incorrects″ se produisent lorsque les anneaux définissant des trous dans le polygone sont aussi orientés dans le sens des aiguilles d'un montre. En effet, cela entraîne des chevauchements des zones intérieures.

Exemple d'instance de polygone :

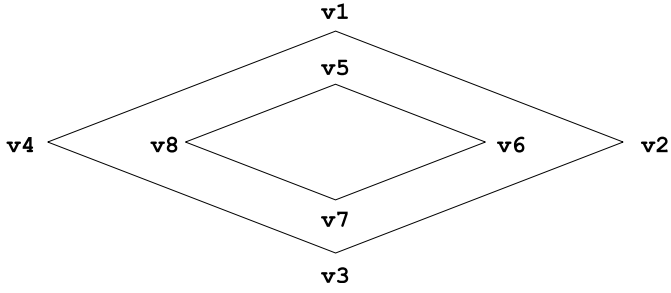


Figure 40. Polygone doté d'un trou et de quatre sommets

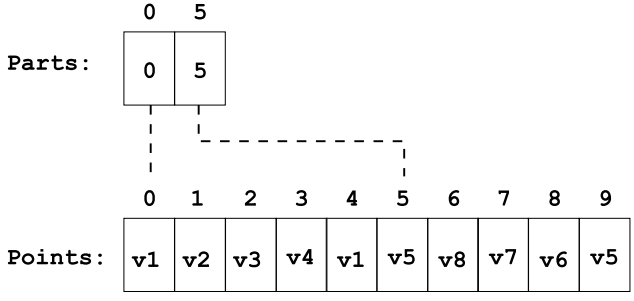


Figure 41. Contenu du train d'octet du polygone. NumParts est égal à 2 et NumPoints à 10. Vous remarquerez que l'ordre des points du polygone (trou) du ″beignet″ est inversé.

Tableau 69. Contenu du train d'octets d'un polygone

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	5	Integer	1	Little
Octet 4	Box	Box	Double	4	Little
Octet 36	NumParts	NumParts	Integer	1	Little
Octet 40	NumPoints	NumPoints	Integer	1	Little
Octet 44	Parts	Parts	Integer	NumParts	Little

Tableau 69. Contenu du train d'octets d'un polygone (suite)

Position	Zone	Valeur	Type	Nombre	Ordre
Octet X	Points	Points	Point	NumPoints	Little

Remarque : X = 44 + 4 * NumParts.

Types de formes mesurées dans l'espace XY

PointM

Un PointM consiste en une paire de coordonnées à double précision dans l'ordre X, Y, plus une mesure M.

Tableau 70. Contenu du train d'octets de PointM

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	21	Integer	1	Little
Octet 4	X	X	Double	1	Little
Octet 12	Y	Y	Double	1	Little
Octet 20	M	M	Double	1	Little

MultiPointM

Les zones d'un MultiPointM sont les suivantes :

Box Boîte englobante associée au MultiPointM et stockée dans l'ordre suivant : Ymin, Ymin, Xmax, Ymax.

NumPoints
Nombre de points.

Points Tableau de points de longueur NumPoints.

NumMs
Nombre de mesures qui suivent. NumMs ne peut avoir que deux valeurs 0 si aucune mesure ne suit cette zone, ou il peut être égal à NumPoints, si des mesures sont spécifiées.

M Range
Mesures minimale et maximale associées au MultiPointM et stockées dans l'ordre suivant : Mmin, Mmax.

M Array
Tableau des mesures de longueur NumPoints.

Tableau 71. Contenu du train d'octets de MultiPointM

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	28	Integer	1	Little
Octet 4	Box	Box	Double	4	Little
Octet 36	NumPoints	NumPoints	Integer	1	Little
Octet 40	Points	Points	Point	NumPoints	Little
Octet X	NumMs	NumMs	Integer	1	Little
Octet X+4*	Mmin	Mmin	Double	1	Little
Octet X+12*	Mmax	Mmax	Double	1	Little
Octet X+20*	Marray	Marray	Double	NumPoints	Little

Remarques :

1. X = 40 + (16 * NumPoints)
2. * facultatif

PolyLineM

Une PolyLineM consiste en une ou plusieurs parties. Une partie est une suite connectée de plusieurs points (deux ou plus). Les parties peuvent être ou non connectées les unes aux autres. Des parties peuvent ou non se couper.

Les zones d'une PolyLineM sont les suivantes :

Box Boîte englobante associée au PolyLineM et stockée dans l'ordre suivant : Ymin, Ymin, Xmax, Ymax.

NumParts
Nombre de parties de la PolyLineM.

NumPoints
Nombre total de points pour toutes les parties.

Parts Tableau de longueur NumParts. Pour chaque anneau, stocke l'indice de son premier point dans le tableau des points. Les indices de tableau sont indiqués par rapport à 0.

Points Tableau de longueur NumPoints. Les points de chaque partie de la PolyLineM sont stockés d'extrémité en extrémité. Les points de la deuxième partie suivent ceux de la première, etc. Le tableau des parties contient l'indice de tableau du point de départ de chaque partie. Il n'existe pas de délimiteur entre parties dans le tableau des points.

NumMs
Nombre de mesures qui suivent. NumMs ne peut avoir que deux

valeurs 0 si aucune mesure ne suit cette zone, ou il peut être égal à NumPoints, si des mesures sont spécifiées.

M Range
Mesures minimale et maximale associées au PolyLineM et stockées dans l'ordre suivant : Mmin, Mmax.

M Array
Tableau de longueur NumPoints. Les mesures de chaque partie de la PolyLineM sont stockées d'extrémité en extrémité. Les mesures de la deuxième partie suivent ceux de la première, etc. Le tableau des parties contient l'indice de tableau du point de départ de chaque partie. Il n'existe pas de délimiteur entre parties dans le tableau des mesures.

Tableau 72. Contenu du train d'octets de PolyLineM

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	13	Integer	1	Little
Octet 4	Box	Box	Double	4	Little
Octet 36	NumParts	NumParts	Integer	1	Little
Octet 40	NumPoints	NumPoints	Integer	1	Little
Octet 44	Parts	Parts	Integer	NumParts	Little
Octet X	Points	Points	Point	NumPoints	Little
Octet Y	NumMs	NumMs	Integer	1	Little
Octet Y+4*	Mmin	Mmin	Double	1	Little
Octet Y+12*	Mmax	Mmax	Double	1	Little
Octet Y+20*	Marray	Marray	Double	NumPoints	Little

Remarques :

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * facultatif

PolygonM

Un PolygonM consiste en un nombre d'anneaux. Un anneau est une boucle fermée qui ne forme pas d'intersection avec elle-même. Vous remarquerez que les intersections sont calculées dans l'espace XY, et *non* dans l'espace XYM. Un PolygonM peut contenir plusieurs anneaux externes. Les anneaux d'un PolygonM sont appelées parties.

Les zones d'une PolygonM sont les suivantes :

Box Boîte englobante associée au PolygonM et stockée dans l'ordre suivant : Ymin, Ymin, Xmax, Ymax.

NumParts
Nombre d'anneaux compris dans le PolygonM.

NumPoints
Nombre total de points pour tous les anneaux.

Parts Tableau de longueur NumParts. Stocke, pour chaque anneau, l'indice de son premier point dans le tableau des points. Les indices de tableau sont indiqués par rapport à 0.

Points Tableau de longueur NumPoints. Les points de chaque anneau du PolyligonM sont stockés d'extrémité en extrémité. Les points du deuxième anneau suivent ceux du premier, etc. Le tableau des parties contient l'indice de tableau du point de départ de chaque anneau. Il n'existe pas de délimiteur entre anneaux dans le tableau des points.

NumMs
Nombre de mesures qui suivent. NumMs ne peut avoir que deux valeurs 0 si aucune mesure ne suit cette zone, ou il peut être égal à NumPoints, si des mesures sont spécifiées.

M Range
Mesures minimale et maximale associées au PolygonM et stockées dans l'ordre suivant : Mmin, Mmax.

M Array
Tableau de longueur NumPoints. Les mesures de chaque anneau du PolyligonM sont stockés d'extrémité en extrémité. Les mesures du deuxième anneau suivent celles du premier, etc. Le tableau des parties contient l'indice de tableau de la mesure de départ de chaque anneau. Il n'existe pas de délimiteur entre anneaux dans le tableau des mesures.

Remarques importantes sur les formes PolygonM :

- Les anneaux sont fermés (le premier sommet d'un anneau est aussi le dernier).
- L'ordre des anneaux dans le tableau des points n'a pas d'importance.

Tableau 73. Contenu du train d'octets de PolygonM

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	15	Integer	1	Little
Octet 4	Box	Box	Double	4	Little
Octet 36	NumParts	NumParts	Integer	1	Little
Octet 40	NumPoints	NumPoints	Integer	1	Little
Octet 44	Parts	Parts	Integer	NumParts	Little
Octet X	Points	Points	Point	NumPoints	Little

Tableau 73. Contenu du train d'octets de PolygonM (suite)

Position	Zone	Valeur	Type	Nombre	Ordre
Octet Y	NumMs	NumMs	Integer	1	Little
Octet Y+4*	Mmin	Mmin	Double	1	Little
Octet Y+12*	Mmax	Mmax	Double	1	Little
Octet Y+20*	Marray	Marray	Double	NumPoints	Little

Remarques :

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * facultatif

Types de formes dans l'espace XYZ

PointZ

Un PointZ consiste en un triplet de coordonnées à double précision dans l'ordre X, Y, Z, plus une mesure M.

Tableau 74. Contenu du train d'octets du PointZ

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	11	Integer	1	Little
Octet 4	X	X	Double	1	Little
Octet 12	Y	Y	Double	1	Little
Octet 20	Z	Z	Double	1	Little
Octet 28	Measure	M	Double	1	Little

MultiPointZ

Un MultiPointZ représente un ensemble de PointZ, comme suit :

- La boîte englobante est stockée dans l'ordre suivant : Ymin, Ymin, Xmax, Ymax.
- La plage Z englobante est stockée dans l'ordre Zmin, Zmax. La plage M englobante est stockée dans l'odre Mmin, Mmax.

Tableau 75. Contenu du train d'octets du MultiPointZ

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	18	Integer	1	Little
Octet 4	Box	Box	Double	4	Little
Octet 36	NumPoints	NumPoints	Integer	1	Little
Octet 40	Points	Points	Point	NumPoints	Little
Octet X	Zmin	Zmin	Double	1	Little

Tableau 75. Contenu du train d'octets du MultiPointZ (suite)

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 36	Zmax	Zmax	Double	1	Little
Octet X+16	Zarray	Zarray	Double	NumPoints	Little
Octet Y	NumMs	NumMs	Integer	1	Little
Octet Y+4*	Mmin	Mmin	Double	1	Little
Octet Y+12*	Mmax	Mmax	Double	1	Little
Octet Y+20*	Marray	Marray	Double	NumPoints	Little

Remarques :

1. X = 40 + (16 * NumPoints); Y = X + 16 + (8 * NumPoints)
2. * facultatif

PolyLineZ

Un fichier shape PolyLineZ consiste en une ou plusieurs parties. Une partie est une suite connectée de plusieurs points (deux ou plus). Les parties peuvent être ou non connectées les unes aux autres. Des parties peuvent ou non se couper.

Les zones d'une PolyLineZ sont les suivantes :

Box Boîte englobante associée à la PolyLineZ et stockée dans l'ordre suivant : Ymin, Ymin, Xmax, Ymax.

NumParts
Nombre de parties de la PolyLineZ.

NumPoints
Nombre total de points pour toutes les parties.

Parts Tableau de longueur NumParts. Pour chaque anneau, stocke l'indice de son premier point dans le tableau des points. Les indices de tableau sont indiqués par rapport à 0.

Points Tableau de longueur NumPoints. Les points de chaque partie de la PolyLineZ sont stockés d'extrémité en extrémité. Les points de la deuxième partie suivent ceux de la première, etc. Le tableau des parties contient l'indice de tableau du point de départ de chaque partie. Il n'existe pas de délimiteur entre parties dans le tableau des points.

Z Range
Valeurs Z minimales et maximales assocées à la PolyLineZ stockées dans l'ordre suivant : Zmin, Zmax.

Z Array

Tableau de longueur NumPoints. Les valeurs Z de chaque partie de la PolyLineZ sont stockés d'extrémité en extrémité. Les valeurs Z de la deuxième partie suivent ceux de la première, etc. Le tableau des parties contient l'indice de tableau du point de départ de chaque partie. Il n'existe pas de délimiteur entre parties dans le tableau des coordonnées Z.

NumMs

Nombre de mesures qui suivent. NumMs ne peut avoir que deux valeurs 0 si aucune mesure ne suit cette zone, ou il peut être égal à NumPoints, si des mesures sont spécifiées.

M Range

Mesures minimale et maximale associées au PolyLineZ et stockées dans l'ordre suivant : Mmin, Mmax.

M Array

Tableau de longueur NumPoints. Les mesures de chaque partie de la PolyLineZ sont stockées d'extrémité en extrémité. Les mesures de la deuxième partie suivent ceux de la première, etc. Le tableau des parties contient l'indice de tableau de la mesure de départ de chaque partie. Il n'existe pas de délimiteur entre parties dans le tableau des mesures.

Tableau 76. Contenu du train d'octets du PolyLineZ

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	13	Integer	1	Little
Octet 4	Box	Box	Double	4	Little
Octet 36	NumParts	NumParts	Integer	1	Little
Octet 40	NumPoints	NumPoints	Integer	1	Little
Octet 44	Parts	Parts	Integer	NumParts	Little
Octet X	Points	Points	Point	NumPoints	Little
Octet Y	Zmin	Zmin	Double	1	Little
Octet Y+8	Zmax	Zmax	Double	1	Little
Octet Y+16	Zarray	Zarray	Double	NumPoints	Little
Octet Z	NumMs	NumMs	Integer	1	Little
Octet Z+4*	Mmin	Mmin	Double	1	Little
Octet Z+12*	Mmax	Mmax	Double	1	Little
Octet Z+20*	Marray	Marray	Double	NumPoints	Little

Remarques :

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints), Z = Y + 16 + (8 * NumPoints)
2. * facultatif

PolygonZ

Un PolygonZ consiste en un nombre d'anneaux. Un anneau est une boucle fermée qui ne forme pas d'intersection avec elle-même. Un PolygonZ peut contenir plusieurs anneaux externes. Les anneaux d'un PolygonZ sont appelées parties.

Les zones d'un PolygonZ sont les suivantes :

Box Boîte englobante associée au PolygonZ et stockée dans l'ordre suivant : Ymin, Ymin, Xmax, Ymax.

NumParts
Nombre d'anneaux compris dans le PolygonZ.

NumPoints
Nombre total de points pour tous les anneaux.

Parts Tableau de longueur NumParts. Stocke, pour chaque anneau, l'indice de son premier point dans le tableau des points. Les indices de tableau sont indiqués par rapport à 0.

Points Tableau de longueur NumPoints. Les points de chaque anneau du PolyligonZ sont stockés d'extrémité en extrémité. Les points du deuxième anneau suivent ceux du premier, etc. Le tableau des parties contient l'indice de tableau du point de départ de chaque anneau. Il n'existe pas de délimiteur entre anneaux dans le tableau des points.

Z Range
Valeurs Z minimales et maximales associées à l'arc stockées dans l'ordre suivant : Zmin, Zmax.

Z Array
Tableau de longueur NumPoints. Les valeurs Z de chaque anneau du PolyligonZ sont stockés d'extrémité en extrémité. Les valeurs Z de la deuxième partie suivent ceux de la première, etc. Le tableau des parties contient l'indice de tableau de la valeur Z de départ de chaque anneau. Il n'existe pas de délimiteur entre anneaux dans le tableau des valeurs Z.

NumMs
Nombre de mesures qui suivent. NumMs ne peut avoir que deux valeurs 0 si aucune mesure ne suit cette zone, ou il peut être égal à NumPoints, si des mesures sont spécifiées.

M Range

Mesures minimale et maximale associées au PolygonZ et stockées dans l'ordre suivant : Mmin, Mmax.

M Array

Tableau de longueur NumPoints. Les mesures de chaque anneau du PolyligonZ sont stockés d'extrémité en extrémité. Les mesures du deuxième anneau suivent celles du premier, etc. Le tableau des parties contient l'indice de tableau de la mesure de départ de chaque anneau. Il n'existe pas de délimiteur entre anneaux dans le tableau des mesures.

Remarques importantes sur les formes de type :

- Les anneaux sont fermés (le premier sommet d'un anneau est aussi le dernier).
- L'ordre des anneaux dans le tableau des points n'a pas d'importance.

Tableau 77. Contenu du train d'octets de PolygonZ

Position	Zone	Valeur	Type	Nombre	Ordre
Octet 0	Shape Type	15	Integer	1	Little
Octet 4	Box	Box	Double	4	Little
Octet 36	NumParts	NumParts	Integer	1	Little
Octet 40	NumPoints	NumPoints	Integer	1	Little
Octet 44	Parts	Parts	Integer	NumParts	Little
Octet X	Points	Points	Point	NumPoints	Little
Octet Y	Zmin	Zmin	Double	1	Little
Octet Y+8	Zmax	Zmax	Double	1	Little
Octet Y+16	Zarray	Zarray	Double	NumPoints	Little
Octet Z	NumMs	NumMs	Integer	1	Little
Octet Z+4*	Mmin	Mmin	Double	1	Little
Octet Z+12*	Mmax	Mmax	Double	1	Little
Octet Z+20*	Marray	Marray	Double	NumPoints	Little

Partie 3. Annexes

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevets couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM EMEA Director of Licensing
IBM Europe Middle-East Africa
Tour Descartes
La Défense 5
2, avenue Gambetta
92066 Paris-La Défense Cedex
France

Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7
Canada

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japon

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales : LE PRESENT DOCUMENT EST LIVRE «EN L'ETAT». IBM DECLINE TOUTE RESPONSABILITE, EXPRESSE OU IMPLICITE, RELATIVE AUX INFORMATIONS QUI Y SONT CONTENUES, Y COMPRIS EN CE QUI CONCERNE LES GARANTIES DE QUALITE MARCHANDE OU D'ADAPTATION A VOS BESOINS. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Il est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut modifier sans préavis les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Ces informations peuvent être soumises à des conditions particulières prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux termes du Contrat sur les produits et services IBM, des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Ce document peut contenir des exemples de données et des rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Le présent logiciel peut contenir des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquelles ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes.

Toute copie totale ou partielle de ces programmes exemples et des oeuvres qui en sont dérivées doit comprendre une notice de copyright, libellée comme suit :

© (nom de votre société) (année). Des segments de code sont dérivés des Programmes exemples d'IBM Corp. © Copyright IBM Corp. _indiquez l'année ou les années_. All rights reserved.

Marques

Les termes qui suivent, accompagnés d'un astérisque (*) dans le document, sont des marques d'International Business Machines Corporation dans certains pays.

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

Les termes qui suivent sont des marques d'autres sociétés :

Microsoft, Windows et Windows NT sont des marques de Microsoft Corporation dans certains pays.

Java, ou toutes les marques et logos incluant Java, et Solaris sont des marques de Sun Microsystems, Inc.

Tivoli et NetView sont des marques de Tivoli Systems Inc. dans certains pays.

UNIX est une marque enregistrée aux Etats-Unis et/ou dans d'autres pays et utilisée avec l'autorisation exclusive de la société X/Open Company Limited.

D'autres sociétés sont propriétaires des autres marques, noms de produits ou logos accompagnés de deux astérisques (**) qui pourraient apparaître dans ce document.

Index

E

F

G

I

J

L

M

O

P

R

S

T

U

V

W

Z

Comment prendre contact avec IBM

Si votre question est d'ordre technique, étudiez tout d'abord les solutions présentées dans le manuel *Troubleshooting Guide* avant de prendre contact avec le Service clients DB2. Ce manuel indique les informations susceptibles d'aider le Service clients à mieux répondre à vos besoins.

Pour obtenir des informations ou commander des produits DB2 avant de prendre contact avec le Service clients DB2 Universal Database, prenez contact avec votre partenaire commercial IBM.

Aux États-Unis, composez l'un des numéros suivants :

- 1-800-237-5511 pour obtenir le Service clients,
- 1-888-426-4343 pour connaître les options de service disponibles.

Infos produit

Aux États-Unis, composez l'un des numéros ci-après.

- Pour commander des produits ou obtenir des informations générales, composez le 1-800-IBM-CALL (1-800-426-2255) ou 1-800-3IBM-OS2 (1-800-342-6672).
- Pour commander des manuels, composez le 1-800-879-2755.

http://www.ibm.com/software/data/
Les pages DB2 World Wide Web fournissent des informations sur DB2, des descriptions de produit, les programmes de formation et d'autres informations.

http://www.ibm.com/software/data/db2/library/
DB2 Product and Service Technical Library permet d'accéder à des forums Q&A (questions/réponses), d'obtenir des correctifs et les dernières informations techniques sur DB2.

Remarque : (Il est possible que ces informations ne soient disponibles qu'en anglais.)

http://www.elink.ibmlink.ibm.com/pbl/pbl/
Le site Web de commande internationale de manuels fournit les informations correspondantes.

http://www.ibm.com/education/certify/
Le programme Professional Certification Program du site Web IBM fournit des informations sur les tests de certification concernant différents produits IBM, dont DB2.

ftp.software.ibm.com
Établissez une connexion anonyme. Des démonstrations, des correctifs, des informations et des outils associés à DB2 ou à des produits connexes sont disponibles dans le répertoire /ps/products/db2.

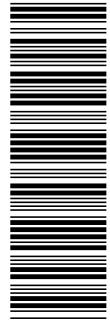
comp.databases.ibm-db2, bit.listserv.db2-l
Ces newsgroups sont accessibles à tous ceux qui souhaitent partager leurs expériences sur les produits DB2.

Sur Compuserve : GO IBMDB2
Exécutez cette commande pour accéder aux forums IBM DB2. Tous les produits DB2 sont pris en charge sur ces forums.

En dehors des Etats-Unis, pour savoir comment prendre contact avec IBM, consultez l'annexe A du manuel *IBM Software Support Handbook*. Pour accéder à ce document, allez sur le site Web : http://www.ibm.com/support/, puis effectuez une recherche sur le mot clé «handbook».

Remarque : Dans certains pays, les distributeurs agréés peuvent contacter leur centre d'assistance au lieu de prendre contact avec le centre de support IBM.

Référence: CT7C0FR



CT7C0FR

SC11-1684-00

Spine information:

IBM DB2 Extension Spatiale

DB2 Extension Spatiale - Guide d'utilisation et de référence

Version 7