IBM DB2 Spatial Extender

Benutzer- und Referenzhandbuch

Version 7

SC12-2894-00

IBM DB2 Spatial Extender

Benutzer- und Referenzhandbuch

Version 7

SC12-2894-00

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter „Bemerkungen" auf Seite 331 gelesen werden.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 Spatial Extender User's Guide and Reference,
IBM Form SC27-0701-00

herausgegeben von International Business Machines Corporation, USA

Herausgegeben von:
SW NLS Center
Kst. 2877
April 2000

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Zu diesem Handbuch

Dieses Buch ist in zwei Teile gegliedert. Der erste Teil enthält Informationen zum Konzept des DB2 Spatial Extender und erläutert die Installation, Konfiguration, Administration und Programmierung des DB2 Spatial Extender auf Windows NT- und AIX-Systemen. Der zweite Teil umfaßt Referenzinformationen zu gespeicherten Prozeduren, Geometrien, Funktionen, Nachrichten und Katalogsichten, die Sie mit DB2 Spatial Extender verwenden.

Zielgruppe

Dieses Buch wendet sich an Administratoren, die die räumliche Umgebung einrichten, und an Anwendungsprogrammierer, die Anwendungen mit räumlichen Daten entwickeln.

Konventionen

In diesem Buch werden die folgenden Hervorhebungskonventionen verwendet:

Fettdruck
: Kennzeichnet Befehle und Steuerzeichen der grafischen Benutzerschnittstelle (GUI), z. B. Feldnamen, Ordnernamen oder Menüauswahlen.

`Monospace-Schrift`
: Kennzeichnet Beispiele zur Codierung von eingegebenem Text.

Kursivschrift
: Kennzeichnet Variablen, die Sie durch einen Wert ersetzen müssen. Kursivschrift dient außerdem der Kennzeichnung von Handbuchtiteln und zum Hervorheben von Wörtern.

GROSSBUCHSTABEN
: Kennzeichnet SQL-Schlüsselwörter und Namen von Objekten (z. B. Tabellen, Sichten und Server).

Anmerkungen einsenden

Ihre Rückmeldungen helfen IBM, Informationen noch besser bereitzustellen. Senden Sie Ihre Anmerkungen zu diesem Handbuch oder anderen Komponenten der DB2-Dokumentation an uns ein. Sie können Ihre Kommentare auf zwei Arten einsenden:

- Senden Sie Ihre Kommentare über das Web. Sie können hierbei das Online-Kommentarformular IBM Data Management unter http://www.ibm.com/software/data/rcf verwenden.
- Senden Sie Ihre Kommentare per E-Mail an comments@vnet.ibm.com. Geben Sie dabei ggf. den Namen des Produkts, die Versionsnummer des Produkts sowie den Namen und die Teilenummer des Buchs an. Wenn Sie Kommentare zu bestimmten Textstellen haben, geben Sie außerdem die Position des entsprechenden Texts an (z. B. Kapitel und Abschnittsnamen, Tabellennummer, Seitennummer oder den Namen eines Hilfeabschnitts) an.

Teil 1. DB2 Spatial Extender verwenden

Kapitel 1. Informationen zum DB2 Spatial Extender

Dieses Kapitel stellt den DB2 Spatial Extender vor. Es beschreibt die Aufgabe des DB2 Spatial Extender sowie die von ihm verarbeiteten Daten und zeigt, wie der DB2 Spatial Extender verwendet wird. Das Kapitel schließt mit einem kurzen Überblick über den Rest des Handbuchs.

Der Zweck des DB2 Spatial Extender

Mit dem DB2 Spatial Extender können Sie ein *geographisches Informationssystem* (GIS) erstellen: einen Komplex aus Objekten, Daten und Anwendungen, der das Generieren und Analysieren räumlicher Informationen zu geographischen Merkmalen ermöglicht. *Geographische Merkmale* umfassen die Objekte, die die Oberfläche der Erde bilden oder sich darauf befinden. Diese Objekte bilden sowohl die natürliche Umgebung (beispielsweise Flüsse, Wälder, Berge oder Wüsten) als auch Kulturland (Städte, Wohngebiete, Gewerbegebiete und Begrenzungen bzw. Grenzen).

Räumliche Informationen umfassen beispielsweise folgende Faktoren:

- Standort geographischer Merkmale in Relation zu ihrer Umgebung (z. B. Standorte von Krankenhäusern in Städten oder die Nähe von Wohngebieten zu Erdbebengebieten.)
- Die Beziehung zwischen verschiedenen geographischen Merkmalen (beispielsweise Informationen darüber, daß sich ein Flußsystem in einer bestimmten Region befindet, oder daß bestimmte Brücken in dieser Region über die Zuflüsse dieses Flußsystems führen)
- Maße, die für ein oder mehrere geographische Merkmale gelten (beispielsweise die Entfernung zwischen einem Bürogebäude und dem Rand des Grundstücks oder der Umkreis eines Vogelschutzgebiets)

Räumliche Informationen können, für sich alleine oder in Verbindung mit den Ausgabedaten eines traditionellen relationalen Datenbankverwaltungssystems (Relational Database Management System, RDBMS), die Konzeption von Projekten unterstützen und Unternehmens- und strategische Entscheidungen erleichtern. Der Leiter einer Sozialbehörde muß beispielsweise überprüfen, welche Antragsteller und Empfänger von Unterstützungsleistungen tatsächlich im Zuständigkeitsbereich seiner Behörde wohnen. DB2 Spatial Extender kann diese Informationen aus der Angabe des Zuständigkeitsbereichs und den Adressen der Personen ableiten.

Oder der Besitzer einer Restaurantkette möchte in anderen Orten der Umgebung weitere Filialen eröffnen. Zum Ermitteln geeigneter Standorte muß er sich Fragen wie die folgenden stellen: Wo in diesen Städten sind die typischen Gäste für meine Restaurants konzentriert? Wo sind die wichtigen Zufahrtsstraßen? Wo ist die Kriminalität am niedrigsten? Wo sind die Restaurants der Konkurrenz? DB2 Spatial Extender kann räumliche Informationen in visuellen Anzeigen erstellen, die diese Fragen beantworten. Das zugrundeliegende RDBMS kann Kennzeichnungen und erläuternde Texte für diese Anzeigen bereitstellen.

Dieses Handbuch enthält noch weitere Beispiele zur Verwendung von DB2 Spatial Extender, insbesondere in „Kapitel 7. Räumliche Informationen abrufen und analysieren" auf Seite 57, „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61 und „Kapitel 14. Räumliche Funktionen für SQL-Abfragen" auf Seite 169.

Daten, die geographische Merkmale darstellen

Dieser Abschnitt bietet einen Überblick über die Daten, die Sie generieren, speichern und bearbeiten, um räumliche Informationen zu erhalten. Folgende Themen werden hierbei behandelt:

- Wie Daten geographische Merkmale darstellen
- Das Wesen der räumlichen Daten
- Möglichkeiten zum Erstellen räumlicher Daten

Wie Daten geographische Merkmale darstellen

In DB2 Spatial Extender kann ein geographisches Merkmal als Zeile in einer Tabelle dargestellt werden oder als Teil einer solchen Zeile. Betrachten Sie beispielsweise zwei der in „Der Zweck des DB2 Spatial Extender" auf Seite 3 erwähnten geographischen Merkmale, Gewerbegebiete und Wohngebiete. In Abb. 1 auf Seite 5 steht jede Zeile der Tabelle BRANCHES für eine Zweigstelle einer Bank. Als Variation dazu steht jede Zeile der Tabelle CUSTOMERS in Abb. 1 auf Seite 5 als Ganzes für einen Kunden der Bank. Ein Teil jeder Zeile — insbesondere die Zellen mit einer Kundenadresse — kann jedoch als Wohnort des Kunden betrachtet werden.

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

Abbildung 1. Tabellenzeile, die ein geographisches Merkmal darstellt; Tabellenzeile, deren Adreßdaten ein geographisches Merkmal darstellt. Die Datenzeile in der Tabelle BRANCHES steht für eine Zweigstelle einer Bank. Die Zellen für Adreßdaten in der Tabelle CUSTOMERS stehen für den Wohnort des Kunden. Die Namen und Adressen in beiden Tabellen sind frei erfunden.

Die Tabellen in Abb. 1 enthalten Daten, die die Zweigstellen und Kunden der Bank kennzeichnen. Solche Daten werden als *attributive Daten* bezeichnet.

Eine Untermenge der attributiven Daten — die Werte, die die Adresse der Zweigstelle und des Kunden angeben — können in Werte umgesetzt werden, die räumliche Daten liefern. Das Beispiel in Abb. 1 zeigt die Zweigstellenadresse 92467 Airzone Blvd., San Jose CA 95141; eine Kundenadresse lautet 9 Concourt Circle, San Jose CA 95141. DB2 Spatial Extender kann diese Adressen in Werte umsetzen, die angeben, wo sich die Zweigstelle und der Wohnort des Kunden in Relation zu der Umgebung befinden. Abb. 2 zeigt die Tabellen BRANCHES und CUSTOMERS mit den neuen Spalten, die solche Werte aufnehmen können.

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

Abbildung 2. Tabellen mit hinzugefügten Spalten für räumliche Daten. In jeder Tabelle wird die Spalte LOCATION die Koordinaten zu der jeweiligen Adresse enthalten.

Wenn Adressen und ähnliche Kennungen als Ausgangspunkt für räumliche Informationen dienen, werden sie als *Quellendaten* bezeichnet. Da die abgeleiteten Werte räumliche Informationen liefern, werden sie als *räumliche Daten* bezeichnet. Der nächste Abschnitt beschreibt die räumlichen Daten und stellt die ihnen zugeordneten Datentypen vor.

Das Wesen der räumlichen Daten

Viele räumliche Daten bestehen aus Koordinaten. Eine *Koordinate* ist eine Zahl, die eine Position in Relation zu einem Bezugspunkt angibt. Breitengrade sind beispielsweise Koordinaten, die die Position relativ zum Äquator angeben. Längengrade sind Koordinaten, die die Position relativ zum Greenwich-Längengrad angeben. Die Position des Yellowstone Nationalparks ist somit durch seinen Breitengrad (44,45 Grad nördlich vom Äquator) und seinen Längengrad (110,40 Grad westlich des Greenwich-Längengrads) festgelegt.

Breiten- und Längengrade, ihre Bezugspunkte und andere zugeordnete Parameter werden zusammen als *Koordinatensystem* bezeichnet. Es gibt auch Koordinatensysteme, die auf anderen Werten basieren als auf Längen- und Breitengraden. Diese Koordinatensysteme haben eigene Maßeinheiten zur Position, eigene Bezugspunkte und weitere eindeutige Parameter.

Das einfachste räumliche Datenelement besteht aus zwei Koordinaten, die die Position eines einzelnen geographischen Merkmals definieren. (Ein *Datenelement* ist der Wert bzw. sind die Werte in einer Zelle einer relationalen Tabelle.) Ein umfangreicheres räumliches Datenelement besteht aus mehreren Koordinaten, die einen linearen Pfad wie beispielsweise eine Straße oder einen Fluß definiert. Eine dritte Art besteht aus Koordinaten, die den Umriß eines Gebiets definieren, beispielsweise den Rand eines Grundstücks oder eines Überschwemmungsgebiets. Diese und andere Arten von DB2 Spatial Extender unterstützte räumlichen Datenelemente werden in „Kapitel 13. Geometrien und zugeordnete räumliche Funktionen” auf Seite 129 ausführlich beschrieben.

Jedes räumliche Datenelement ist ein Exemplar eines Typ von räumlichen Daten. Der Datentyp für zwei Koordinaten, die die Position eines Standorts kennzeichnen, ist ST_Point; der Datentyp für Koordinaten, die lineare Pfade definieren, ist ST_LineString; und der Datentyp für Koordinaten, die einen Umfang definieren, ist ST_Polygon. Diese Typen sowie andere Datentypen für räumliche Daten sind strukturierte Typen, die zu einer einzigen Hierarchie gehören. Einen Überblick über die Hierarchie finden Sie im Abschnitt „Informationen zu Typen von räumlichen Daten” auf Seite 36.

Woher räumliche Daten stammen

Sie können räumliche Daten auf folgende Arten erhalten:

- Durch Ableiten aus attributiven Daten
- Durch Ableiten aus anderen räumlichen Daten
- Durch Importieren

Attributive Daten als Quellendaten verwenden

DB2 Spatial Extender kann räumliche Daten aus attributiven Daten ableiten, beispielsweise aus Adressen (wie beispielsweise in „Wie Daten geographische Merkmale darstellen" auf Seite 4). Dieser Prozeß wird als *Geocodierung* bezeichnet. Zum Anzeigen der betreffenden Sequenz betrachten Sie Abb. 2 auf Seite 5 als „Vorher" und Abb. 3 als „Nachher". Abb. 2 auf Seite 5 zeigt, daß die beiden Tabellen BRANCHES und CUSTOMERS jeweils eine leere Spalte enthalten, die die räumlichen Daten aufnehmen kann. Angenommen, der DB2 Spatial Extender "geocodiert" die Adressen in diesen Tabellen zum Abrufen von Koordinaten dieser Adressen und plaziert diese Koordinaten in den Spalten. Abb. 3 verdeutlicht dieses Ergebnis.

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

Abbildung 3. Tabellen mit räumlichen Daten, die aus Quellendaten abgeleitet wurden. Die Spalte LOCATION in der Tabelle CUSTOMERS enthält Koordinaten, die ein Geocodierer aus der Adresse in den Spalten ADDRESS, CITY, STATE und ZIP abgeleitet hat. Ebenso enthält die Spalte LOCATION in der Tabelle BRANCHES Koordinaten, die der Geocodierer aus der Adresse in den Spalten ADDRESS, CITY, STATE und ZIP dieser Tabelle abgeleitet hat. Dieses Beispiel ist erfunden; es werden simulierte Koordinaten und keine realen Koordinaten dargestellt.

DB2 Spatial Extender verwendet eine als *Geocodierer* bezeichnete Funktion, um attributive Daten in räumliche Daten umzusetzen und diese räumlichen Daten in Tabellenspalten einzutragen. Weitere Informationen zu Geocodierern finden Sie im Abschnitt „Informationen zur Geocodierung" auf Seite 43.

Andere räumliche Daten als Quellendaten verwenden

Räumliche Daten können nicht nur aus attributiven Daten generiert werden, sondern auch aus anderen räumlichen Daten. Die Bank, deren Zweigstellen in der Tabelle BRANCHES definiert sind, möchte beispielsweise wissen, wie viele Kunden ihren Wohnort innerhalb eines Radius von fünf Meilen um die einzelnen Zweigstellen haben. Bevor die Bank diese Informationen aus der Datenbank abrufen kann, muß in der Datenbank die Definition der Zone eingegeben werden, die einen Radius von fünf Meilen um die einzelnen Zweigstellen definiert. Die Funktion ST_Buffer des DB2 Spatial Extender kann eine solche Definition erstellen. Mit den Koordinaten der einzelnen Zweigstellen als Eingabe kann ST_Buffer die Koordinaten generieren, die den Umriß der gewünschten Zonen festlegen. Abb. 4 zeigt die Tabelle BRANCHES mit den von ST_Buffer bereitgestellten Informationen.

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

Abbildung 4. Tabelle, die neue räumliche Daten enthält, die aus vorhandenen räumlichen Daten abgeleitet wurden. Die Koordinaten in der Spalte SALES_AREA wurden mit der Funktion ST_Buffer aus den Koordinaten in der Spalte LOCATION abgeleitet. Wie die Koordinaten in der Spalte LOCATION sind auch diejenigen in der Spalte SALES_AREA nur simuliert und stellen keine realen Koordinaten dar.

Zusätzlich zu ST_Buffer bietet DB2 Spatial Extender verschiedene andere Funktionen, mit denen räumliche Daten aus bereits vorhandenen räumlichen Daten abgeleitet werden können. Beschreibungen von ST_Buffer und diesen weiteren Funktionen finden Sie im Abschnitt „Funktionen, die neue Geometrien aus vorhandenen generieren" auf Seite 159.

Räumliche Daten importieren

Eine dritte Möglichkeit, räumliche Daten zu erhalten, ist das Importieren aus Dateien, die in einem von DB2 Spatial Extender unterstützten Format vorliegen. Eine Beschreibung dieser Formate finden Sie in „Kapitel 16. Dateiformate für räumliche Daten" auf Seite 303. Diese Dateien enthalten Daten, die normalerweise Karten zugeordnet werden: Volkszählungsdaten, Überschwemmungsgebiete, Erdbebenzonen etc. Durch die Verwendung solcher Daten in Verbindung mit den generierten räumlichen Daten können Sie die verfügbaren geographischen Daten erweitern.

Wenn eine Katastrophenschutzbehörde beispielsweise ermitteln will, welchen Risiken ein Wohngebiet ausgesetzt ist, könnte mit ST_Buffer eine Zone um das Gebiet herum definiert werden. Anschließend könnten dann Daten zu Überschwemmungsgebieten und Erdbebenzonen importiert werden, um festzustellen, welche dieser Risiken in der Zone vorliegen.

Ein DB2 Spatial Extender-GIS erstellen und verwenden

Sie erstellen ein DB2 Spatial Extender-GIS durch Einrichten des DB2 Spatial Extender und die Entwicklung von GIS-Projekten in den kombinierten Umgebungen von DB2 Spatial Extender und seiner zugrundeliegenden DB2 RDBMS. Sie verwenden dieses GIS durch die Implementierung dieser Projekte, d. h. durch Generieren und Analysieren der — räumlichen und herkömmlichen — Informationen, für die diese Projekte konzipiert wurden. Hierzu müssen verschiedene Aufgaben ausgeführt werden. Dieser Abschnitt stellt die Schnittstellen vor, mit denen Sie diese Aufgaben durchführen können. Außerdem finden Sie hier einen Überblick über diese Aufgaben und ein Szenario, das diese Aufgaben verdeutlicht.

Schnittstellen zu DB2 Spatial Extender und ihre Funktionalität

In diesem Abschnitt werden die Schnittstellen vorgestellt, über die Sie ein DB2 Spatial Extender-GIS erstellen (Ressourcen definieren, räumliche Daten abrufen, etc.) und verwenden (Informationen zu geographischen Merkmalen generieren und analysieren) können.

Sie können ein DB2 Spatial Extender-GIS wie folgt erstellen:

- Mit den DB2 Spatial Extender-Fenstern und Menüauswahlpunkten des DB2 Control Center. Anleitungen hierzu finden Sie in folgenden Kapiteln:
  - „Kapitel 3. Ressourcen einrichten” auf Seite 25
  - „Kapitel 4. Räumliche Spalten definieren, als Schichten registrieren und Geocodierer aktivieren” auf Seite 35
  - „Kapitel 5. Räumliche Spalten ausfüllen” auf Seite 43
  - „Kapitel 6. Räumliche Indizes erstellen” auf Seite 55
- Durch Ausführen eines Anwendungsprogramms, das gespeicherte Prozeduren des DB2 Spatial Extender aufruft. Hinweise zum Entwickeln eines solchen Programms finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben” auf Seite 61.

- Mit dem Control Center und einem Anwendungsprogramm. Sie können beispielsweise über das Control Center den Standard-Geocodierer aufrufen. Wenn Sie darüber hinaus einen weiteren Geocodierer verwenden wollen, müssen Sie ihn zunächst in DB2 Spatial Extender registrieren, indem Sie die gespeicherte Prozedur db2gse.gse_register_gc in einem Anwendungsprogramm aufrufen. (Informationen zu Geocodierern über den Standardcodierer hinaus finden Sie im Abschnitt „Informationen zur Geocodierung" auf Seite 43. Informationen zu der gespeicherten Prozedur db2gse.gse_register_gc finden Sie im Abschnitt „db2gse.gse_register_gc" auf Seite 94.)
- Mit dem Control Center, einem Anwendungsprogramm oder beiden Oberflächen in Verbindung mit weiteren Schnittstellen. Zum Erstellen einer Tabelle, die die durch eine räumliche Funktion wie beispielsweise einen Geocodierer zu generierenden Daten enthalten soll, können Sie den Befehlszeilenprozessor oder die Control Center-Schnittstellen verwenden.

Sie können ein DB2 Spatial Extender-GIS auf folgende Arten verwenden:

- Durch die graphische Wiedergabe von Daten mit einem Geobrowser; ein Beispiel hierfür ist der vom Environmental Systems Research Institute (ESRI) angebotene ArcExplorer.
- Durch das explizite Übergeben von SQL-Abfragen über das DB2 Control Center oder den Befehlszeilenprozessor
- Durch das Übergeben von SQL-Abfragen aus einem Anwendungsprogramm

Aufgaben zum Erstellen und Verwenden eines DB2 Spatial Extender-GIS

Dieser Abschnitt bietet einen Überblick über die Aufgaben, über die Sie ein DB2 Spatial Extender-GIS erstellen und verwenden. Die Aufgaben, über die Sie das GIS erstellen, umfassen das Einrichten des DB2 Spatial Extender und die Entwicklung von GIS-Projekten. Die Aufgaben zum Verwenden des GIS umfassen das Implementieren der Projekte. Diese Übersicht beginnt mit dem Einrichten des DB2 Spatial Extender; im Anschluß daran werden die Entwicklung und Implementierung einer GIS-Projekts beschrieben. Der Abschnitt schließt mit einem Hinweis darauf, wie sich die in der Übersicht beschriebenen Aufgaben in der täglichen Praxis unterscheiden können.

Den DB2 Spatial Extender einrichten

Das Einrichten des DB2 Spatial Extender umfaßt folgende Punkte:

1. Planen und Vorbereiten (Festlegen, welche GIS-Projekte entwickelt werden sollen, welche Datenbank für den DB2 Spatial Extender aktiviert werden soll, Auswählen der Mitarbeiter für die Verwaltung des DB2 Spatial Extender und die Entwicklung der Projekte, etc.).
2. Installieren des DB2 Spatial Extender.
3. Bereitstellen der Ressourcen zur Unterstützung der GIS-Projekte, z. B.:

   Vom DB2 Spatial Extender bereitgestellte Ressourcen
   Hierzu gehören ein Systemkatalog, Typen von räumlichen Daten, räumliche Funktionen (einschließlich eines Standard-Geocodierers) etc. Das Einrichten dieser Ressourcen wird als *Aktivieren der Datenbank für räumliche Operationen* bezeichnet.

   Von Benutzern, Lieferanten oder beiden entwickelte Geocodierer
   Der Standard-Geocodierer wandelt Adressen in den USA um in räumliche Daten. Ihre Organisation und andere Lieferanten können Geocodierer bereitstellen, die Adressen aus anderen Regionen sowie andere Arten attributiver Daten in räumliche Daten umsetzen.

Anweisungen zum Installieren des DB2 Spatial Extender finden Sie in „Kapitel 2. DB2 Spatial Extender installieren" auf Seite 19. Anweisungen zum Verwenden des Control Center zum Bereitstellen von Ressourcen finden Sie in „Kapitel 3. Ressourcen einrichten" auf Seite 25. Richtlinien zur Verwendung eines Anwendungsprogramms zu diesem Zweck finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61. Ein Szenario, das den Gesamtaufwand beim Einrichten des DB2 Spatial Extender verdeutlicht, finden Sie im Abschnitt „Ein System zur Integration räumlicher und traditioneller Daten" auf Seite 14.

Ein GIS-Projekt entwickeln und implementieren

Die Entwicklung und Implementierung eines GIS-Projekts umfaßt folgende Aufgaben:

1. Planen und Vorbereiten (Projektziele definieren, Ermitteln, welche Tabellen und Daten erforderlich sind, Festlegen der zu verwendenden Koordinatensysteme etc.).
2. Festlegen, welche räumlichen Bezugssysteme verwendet werden sollen. Die Koordinatenwerte umfassen normalerweise positive ganze Zahlen, negative Zahlen sowie Dezimalzahlen. DB2 Spatial Extender muß jedoch alle Koordinatenwerte in Form von ganzen Zahlen speichern. Ein *räumliches Bezugssystem* ist eine Gruppe von Parametern, die definiert, wie negative Zahlen und Dezimalzahlen in einem bestimmten Koordinatensystem in positive ganze Zahlen umgewandelt werden sollen, so daß DB2 Spatial Extender sie speichern kann. Nachdem Sie festgelegt haben, welches Koordinatensystem für eine räumliche Spalte verwendet werden soll, müssen Sie das räumliche Bezugssystem angeben, über das die erforderliche Umwandlung für diese Spalte durchgeführt werden kann. Wenn ein vorhandenes räumliches Bezugssystem Ihren Anforderungen entspricht, können Sie es verwenden; andernfalls können Sie ein solches System erstellen.
3. Definieren einer oder mehrerer Spalten für räumliche Daten, Registrieren der Spalten für DB2 Spatial Extender und Aktivieren eines Geocodierers, der diese Spalten automatisch verwaltet.

   Die Registrierung einer räumlichen Spalte umfaßt ihre Aufzeichnung im DB2 Spatial Extender-Katalog. Von dem Zeitpunkt der Registrierung ab wird die Spalte als *Schicht* bezeichnet, da durch die aus ihr generierten Daten der von Ihrem GIS erstellten virtuellen geographischen Landschaft eine geologische Schicht hinzugefügt wird. Nach der Registrierung können Sie räumliche Operationen mit der Spalte durchführen; Sie können sie beispielsweise ausfüllen und einen räumlichen Index dafür definieren.
4. Ausfüllen räumlicher Spalten:
   - Legen Sie für ein Projekt, das einen Geocodierer erfordert, Parameter für den Geocodierer fest. Führen Sie das Projekt anschließend aus, so daß es in einer einzigen Operation alle verfügbaren Quellendaten geocodiert und die resultierenden Koordinaten in einer Schicht lädt.
   - Importieren Sie die räumlichen Daten, falls dies für das Projekt erforderlich ist.
5. Ermöglichen Sie den Zugriff auf räumliche Spalten. Dies umfaßt insbesondere das Definieren von Indizes, die DB einen schnellen Zugriff auf die räumlichen Daten ermöglichen, sowie das Definieren von Sichten, über die die Benutzer die in Wechselbeziehung zueinander stehenden Daten effizient abrufen können. Nach dem Definieren einer solchen Sicht müssen Sie ihre räumlichen Spalten als Schichten registrieren.
6. Generieren und Analysieren räumlicher Informationen und der dazugehörigen Geschäftsinformationen. Dies umfaßt das Abfragen räumlicher Spal-

ten und der dazugehörigen Attributspalten. Bei solchen Abfragen können Sie DB2 Spatial Extender-Funktionen verwenden, die eine Vielzahl verschiedener Informationen zurückgeben, beispielsweise den Mindestabstand zwischen zwei geographischen Merkmalen oder die Koordinaten, die einen Bereich um ein geographisches Merkmal herum definieren. Informationen zu der Funktion, die solche Koordinaten zurückgibt, ST_Buffer, finden Sie in den Abschnitten „Andere räumliche Daten als Quellendaten verwenden" auf Seite 8 und „ST_Buffer" auf Seite 199. Beispiele zu den Abfragen, die räumliche Funktionen verwenden, finden Sie in „Kapitel 7. Räumliche Informationen abrufen und analysieren" auf Seite 57 und „Kapitel 14. Räumliche Funktionen für SQL-Abfragen" auf Seite 169.

Anweisungen zum Verwenden des Control Center zur Ausführung der Aufgaben bei der Entwicklung eines GIS-Projekts finden Sie in den folgenden Kapiteln:

- „Kapitel 3. Ressourcen einrichten" auf Seite 25
- „Kapitel 4. Räumliche Spalten definieren, als Schichten registrieren und Geocodierer aktivieren" auf Seite 35
- „Kapitel 5. Räumliche Spalten ausfüllen" auf Seite 43
- „Kapitel 6. Räumliche Indizes erstellen" auf Seite 55

Richtlinien zur Verwendung des Control Center bei der Implementierung eines GIS-Projekts finden Sie in „Kapitel 7. Räumliche Informationen abrufen und analysieren" auf Seite 57.

Richtlinien zur Verwendung eines Anwendungsprogramms bei der Entwicklung und Implementierung eines GIS-Projekts finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Ein Szenario, das den Gesamtaufwand verdeutlicht, finden Sie im Abschnitt „Ein Projekt zum Einrichten von Niederlassungen und zum Anpassen von Prämien" auf Seite 15.

Unterschiedliche Gruppen von Aufgaben

Die Gruppe von Aufgaben, die Sie zum Erstellen und Verwenden eines DB2 Spatial Extender-GIS ausführen, kann in Inhalt und Reihenfolge variieren, je nach Ihren Anforderungen und den verwendeten Schnittstellen. Betrachten Sie beispielsweise die Aufgabe zum Definieren von Spalten für räumliche Daten, das Registrieren der Spalten als Schichten und das Aktivieren eines Geocodierers, der diese Spalten automatisch verwaltet. Über das Control Center können Sie diese Ausgaben über ein einziges Fenster zusammen ausführen. Wenn Sie jedoch gespeicherte Prozeduren von einem Programm aus aufrufen, können Sie diese Aufgaben separat ausführen und nach eigenem Ermessen zeitlich steuern.

Szenario: Eine Versicherungsgesellschaft aktualisiert ihr GIS

Dieser Abschnitt zeigt ein Szenario zur Verdeutlichung der Gruppe von Aufgaben, die im vorigen Abschnitt beschrieben wurden.

Die Informationssystemumgebungen der Versicherungsgesellschaft Safe Harbor Real Estate umfaßt ein allgemeines DB2-Datenbanksystem und ein separates GIS-Datenbankverwaltungssystem. Bis zu einem gewissen Grad können Abfragen Kombinationen von Daten aus beiden Systemen liefern. Eine DB2-Tabelle speichert beispielsweise Informationen zum Umsatz, und eine GIS-Tabelle speichert die Standorte der Zweigstellen des Unternehmens. Somit können auch die Standorte der Niederlassungen ermittelt werden, die einen Umsatz einer bestimmten Höhe erzielen. Die Daten aus den beiden Systemen können jedoch nicht integriert werden (die Benutzer können beispielsweise nicht DB2-Spalten mit GIS-Spalten verbinden), und DB2-Dienste wie beispielsweise die Optimierung von Abfragen stehen für das GIS nicht zur Verfügung. Zur Überwindung dieser Nachteile kauft Safe Harbor DB2 Spatial Extender und richtet eine neue GIS-Entwicklungsabteilung ein. In den folgenden Abschnitten wird beschrieben, wie diese Abteilung DB2 Spatial Extender einrichtet und ihr erstes Projekt realisiert.

Ein System zur Integration räumlicher und traditioneller Daten

Die GIS-Abteilung von Safe Harbor geht zum Einrichten des DB2 Spatial Extender wie folgt vor:

1. Die Abteilung bereitet das Einbeziehen des DB2 Spatial Extender in seine DB2-Umgebung vor. Beispiel:
   a. Das Management-Team der Abteilung benennt ein Team für die räumliche Verwaltung mit der Aufgabe, DB2 Spatial Extender zu installieren und zu implementieren. Ein anderes Team für die räumliche Analyse soll räumliche Daten generieren und analysieren.
   b. Da die unternehmerischen Entscheidungen von Safe Harbor hauptsächlich durch die Kundenanforderungen bestimmt sind, beschließt das Management-Team, DB2 Spatial Extender in der Datenbank zu installieren, die Informationen über die Kunden enthält. Die meisten dieser Informationen sind in der Tabelle CUSTOMERS gespeichert.

      Die Mitglieder der GIS-Entwicklungsabteilung nennen diese ausgewählte Datenbank einfach eine *GIS-Datenbank*. Dabei ist ihnen durchaus bewußt, daß diese Datenbank nicht ausschließlich für GIS-Zwecke reserviert ist; sie kann nach wie vor auch von anderen Anwendungen verwendet werden.
2. Das Team für die räumliche Verwaltung installiert den DB2 Spatial Extender.
3. Das Team für die räumliche Verwaltung richtet die für GIS-Projekte erforderlichen Ressourcen ein:

- Das Team stellt über das Control Center die Ressourcen bereit, die die GIS-Datenbank für räumliche Operationen aktiviert. Hierzu gehören der DB2 Spatial Extender-Katalog, Typen von räumlichen Daten, räumliche Funktionen etc.
- Da Safe Harbor seine Aktivitäten auch nach Kanada ausdehnen will, beginnt das Team für die räumliche Verwaltung damit, Angebote von kanadischen Anbietern über Geocodierer einzuholen, die Adressen in Kanada in räumliche Daten umsetzen.

Ein Projekt zum Einrichten von Niederlassungen und zum Anpassen von Prämien

Zur Ausführung ihres ersten GIS-Projekts mit DB2 Spatial Extender geht die GIS-Entwicklungsabteilung wie folgt vor:

1. Die Abteilung bereitet die Entwicklung des Projekts vor, z.B.:
   - Das Management-Team definiert die Ziele für das Projekt:
     – Ermitteln, wo neue Zweigstellen eingerichtet werden sollen
     – Anpassen der Prämien entsprechend der Nähe des Kundenwohnorts zu Gefahrengebieten (Gebieten mit hoher Verkehrsunfallquote, hoher Kriminalitätsrate, Überschwemmungsgebiet, Erdbebengebiet, etc.).
   - Das GIS-Projekt umfaßt Kunden und Niederlassungen in den USA. Das Team für die räumliche Verwaltung beschließt daher die Verwendung folgender Systeme:
     – Koordinatensysteme, die Standorte in den Teilen der USA exakt beschreiben, in denen Safe Harbor präsent ist.
     – Standard-Geocodierer, da dieser speziell für die Geocodierung von Adressen in den USA konzipiert wurde.
   - Das Team für die räumliche Verwaltung legt fest, welche Daten zum Erreichen der Projektziele erforderlich sind und welche Tabellen diese Daten enthalten sollen.
2. Über das Control Center erstellt das Team für die räumliche Verwaltung zwei räumliche Bezugssysteme. Das eine System legt fest, wie Koordinaten, die die Standorte der Niederlassungen definieren, in Datenelemente umgewandelt werden sollen, die der DB2 Spatial Extender speichern kann. Das andere System legt fest, wie Koordinaten, die die Wohnorte der Kunden definieren, in Datenelemente umgewandelt werden sollen, die der DB2 Spatial Extender speichern kann.
3. Über das Control Center definiert das Team für die räumliche Verwaltung die Spalten für die räumlichen Daten, registriert sie als Schichten und aktiviert einen Geocodierer, der diese Spalten automatisch verwaltet:
   - Das Team fügt der Tabelle CUSTOMERS eine Spalte LOCATION hinzu. Die Tabelle enthält bereits Kundenadressen. Der Standard-Geocodierer setzt diese Adressen um in räumliche Daten und lädt diese Daten in eine Spalte LOCATION.

- Das Team erstellt eine Tabelle OFFICES für die Daten, die jetzt in dem separaten GIS enthalten sind. Diese Daten enthalten die Adressen der Zweigstellen von Safe Harbor, räumliche Daten, die über einen Geocodierer aus diesen Adressen abgeleitet wurden, und räumliche Daten, die eine Zone mit einem Radius von fünf Meilen um jede Niederlassung definieren. Die von einem Geocodierer generierten Daten werden in eine Spalte LOCATION geschrieben. Die Daten zur Definition der Zone werden in die Spalte SALES_AREA geschrieben.
- Das Team registriert die beiden LOCATION-Spalten und die beiden SALES_AREA-Spalten als Schichten.
- Das Team aktiviert den Standard-Geocodierer zur automatischen Verwaltung der beiden LOCATION-Spalten.

4. Das Team für die räumliche Verwaltung füllt die Spalte LOCATION der Tabelle CUSTOMER, die gesamte Tabelle OFFICES und eine neue Tabelle HAZARD_ZONES aus:
   - Das Team füllt über das Control Center die Spalte LOCATION der Tabelle CUSTOMER aus:
     a. Das Team weist den Geocodierer an, nur unter der folgenden Bedingung räumliche Daten für eine Adresse in die Spalte LOCATION einzutragen: wenn eine Adresse und ihr Pendant in den Aufzeichnungen der United States Census Bureau zu 100 Prozent übereinstimmen. (Eine Datei der vom Census Bureau bereitgestellten Adressen wird zusammen mit DB2 Spatial Extender ausgeliefert. Bevor der Geocodierer eine Adresse aus den Quellendaten in räumliche Daten umwandeln kann, muß der Geocodierer versuchen, diese Adresse dem entsprechenden Pendant in der Datei zuzuordnen. Die Benutzer geben an, wie hoch der übereinstimmende Prozentsatz sein muß, damit die räumlichen Daten in eine Tabelle geschrieben werden. Dieser Prozentsatz wird als *Genauigkeit* bezeichnet.)
     b. Das Team führt den Geocodierer im Stapelbetrieb aus, damit alle Adressen in der Tabelle in einem Arbeitsgang geocodiert werden können. Zum Leidwesen des Teams weist der Geocodierer ungefähr jede zehnte Adresse zurück!
     c. Das Team vermutet, daß es sich bei den zurückgewiesenen Daten um neue Adressen handelt, für die keine genaue Entsprechung in der Datei des Census Bureau vorhanden ist. Zur Lösung des Problems reduziert das Team die Genauigkeit auf 85.
     d. Das Team führt den Geocodierer erneut im Stapelbetrieb aus. Der Anteil der zurückgewiesenen Adressen sinkt auf ein akzeptables Maß.

- Mit einem von dem separaten GIS bereitgestellten Dienstprogramm lädt das Team die Niederlassungsdaten in eine Datei. Anschließend importiert das Team diese Daten über das Control Center aus der Datei in die neue Tabelle OFFICES.
- Über das Control Center erstellt das Team eine Tabelle HAZARD ZONES, registriert ihre räumlichen Spalten als Schichten und importiert Daten in diese Spalten. Die Daten stammen aus einer Datei, die der Anbieter der Karte zur Verfügung gestellt hat.

5. Über das Control Center ermöglicht das Team für die räumliche Verwaltung den Zugriff auf die neuen Schichten:
   - Das Team erstellt Indizes für diese Schichten.
   - Das Team erstellt eine Sicht, die Spalten aus den Tabellen CUSTOMERS und HAZARD ZONES miteinander verknüpft. Anschließend registriert das Team die räumlichen Spalten der Sichten als Schichten.
6. Das Team für die räumliche Verwaltung führt Abfragen aus, um Informationen abzurufen, mit denen die ursprünglichen Ziele erreicht werden können: Ermitteln, ob neue Zweigstellen eingerichtet werden sollen, und Anpassen der Prämien entsprechend der Nähe der Kunden zu Gefahrengebieten.

Kapitel 2. DB2 Spatial Extender installieren

Dieses Kapitel enthält Anleitungen zum Installieren des DB2 Spatial Extender. Folgende Punkte werden beschrieben:

DB2 Spatial Extender-Konfiguration

Ein DB2 Spatial Extender-System besteht aus einer allgemeinen DB2-Datenbank, DB2 Spatial Extender und einem Geobrowser (z. B. ArcExplorer). Normalerweise befindet sich eine für räumliche Operationen aktivierte Datenbank auf dem Server. Mit Hilfe von Client-Anwendungen können Sie über die gespeicherten Prozeduren des DB2 Spatial Extender und über räumliche Abfragen auf räumliche Daten zugreifen. Darüber hinaus können Sie mit einem Geobrowser auf räumliche Daten zugreifen.

Abb. 5 verdeutlicht die Architektur des DB2 Spatial Extender.

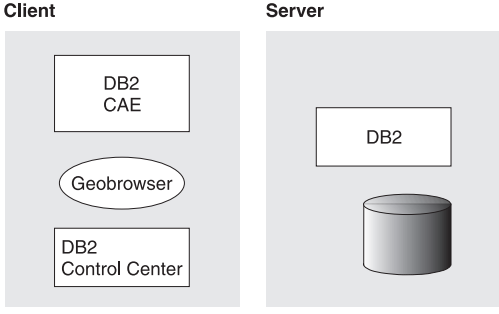


Abbildung 5. Client/Server-Konfiguration

Systemvoraussetzungen

Dieser Abschnitt erläutert die Software- und Hardwarevoraussetzungen für den DB2 Spatial Extender.

Unterstützte Betriebssysteme

DB2 Spatial Extender kann auf folgenden Betriebssystemen installiert werden:

- AIX 4.2 oder höher
- Windows NT 4.0 oder höher mit Service Pack 5

Erforderliche Datenbanksoftware

Bevor Sie den DB2 Spatial Extender installieren, müssen Sie die DB2-Software auf Ihrem System installieren und konfigurieren. Tabelle 1 listet die Voraussetzungen für die Datenbanksoftware für die DB2 Spatial Extender-Client-Komponente und die DB2 Spatial Extender-Server-Komponente auf.

Tabelle 1. Softwarevoraussetzungen

Komponente	Software
Client	DB2 Administration Client, Version 7.1[1]
Server	Entweder: • DB2 Universal Database Enterprise Edition, Version 7.1 • DB2 Universal Database Enterprise – Extended Edition, Version 7.1[2]
	Anmerkungen: 1. Wenn Sie *nicht* vorhaben, mit dem DB2 Control Center zu arbeiten, mit einem Geobrowser auf räumliche Daten zuzugreifen oder das DB2 Spatial Extender-Beispielprogramm zu verwenden, können Sie eine vereinfachte Version des DB2 Administration Client verwenden. 2. Sie können zwar den DB2 Spatial Extender mit DB2 Universal Database Enterprise - Extended Edition verwenden, der räumliche Index kann jedoch nicht über mehrere Knoten hinweg partitioniert werden wie in einer massiven Parallelverarbeitungsumgebung (MPP).

Voraussetzungen für den Plattenspeicherplatz

Tabelle 2 listet die empfohlenen Voraussetzungen für den Plattenspeicherplatz für den DB2 Spatial Extender auf.

Tabelle 2. Voraussetzungen für den Plattenspeicherplatz

DB2 Spatial Extender-Komponente	Plattenspeicherplatz
DB2 Spatial Extender-Server-Bibliothek (einschließlich DB2 Spatial Extender-Server-Bibliothek, Geocodierer-Bezugsdaten und Dokumentation)	600 MB
DB2 Spatial Extender-Client-Unterstützung (einschließlich Daten für das Beispielprogramm)	15 MB

DB2 Spatial Extender installieren

Dieser Abschnitt enthält Informationen, die Sie zur Installation des DB2 Spatial Extender unter den Betriebssystemen Windows NT und AIX benötigen.

Vorbereitungen

Falls Sie dies nicht bereits getan haben, installieren Sie den DB2 Administration Client (Verwaltungs-Tools einschließlich Control Center und Laufzeit-Client) und die Client-Workstation, und installieren Sie DB2 Universal Database Enterprise Edition oder DB2 Universal Database Enterprise - Extended Edition. Anweisungen hierzu finden Sie im entsprechenden Handbuch *Einstieg*.

DB2 Spatial Extender auf Windows NT-Systemen installieren

So installieren Sie den DB2 Spatial Extender auf einem Windows NT-System:

1. Melden Sie sich am System mit einem Benutzernamen an, der die erforderlichen Verwaltungsberechtigungen hat.
2. Schließen Sie alle anderen Programme ab.
3. Legen Sie die CD-ROM in das Laufwerk ein. Daraufhin wird die Installations-Klickstartleiste geöffnet.
4. Wahlfrei: Klicken Sie **Release Notes** an, um die DB2 Spatial Extender-Release-Hinweise zu eventuellen Änderungen beim Installationsprozeß anzuzeigen. Kehren Sie anschließend zur DB2 Spatial Extender-Klickstartleiste zurück.
5. Klicken Sie **Install** an.
6. Führen Sie die Anweisungen des Installationsprogramms aus. Als Anleitung für die weiteren Schritte steht eine Online-Hilfe zur Verfügung. Klikken Sie zum Aufrufen der Online-Hilfe **Help** an, oder drücken Sie die Taste F1.

Nach Abschluß der Installation ist DB2 Spatial Extender im Verzeichnis %DB2PATH% (z. B. c:\sqllib) installiert.

DB2 Spatial Extender auf AIX-Systemen installieren

So installieren Sie DB2 Spatial Extender auf einem AIX-System:

1. Melden Sie sich als Root an.
2. Legen Sie die CD-ROM in das Laufwerk ein.
3. Hängen Sie die CD-ROM an Ihr AIX-System an. Informationen zum Anhängen einer CD-ROM finden Sie im Handbuch *IBM DB2 Universal Database für UNIX Einstieg*.
4. Wechseln Sie in das Verzeichnis, an das die CD-ROM angehängt ist. Geben Sie hierzu den folgenden Befehl ein:

   ```
   cd /cdrom
   ```

   cdrom steht hierbei für das CD-ROM-Laufwerk unter AIX.

5. Geben Sie den Befehl **db2setup** ein, um das DB2-Installationsprogramm zu starten. Das Fenster **Install DB2 Spatial Extender** wird geöffnet.

   Anmerkung: Das DB2-Installationsprogramm arbeitet nach dem Starten sehr langsam, weil es zunächst Ihr System nach Informationen durchsucht.

6. Wählen Sie in der Produktliste des Fensters **Install DB2 Spatial Extender** die zu installierenden Produkte aus, und klicken Sie **OK** an.

   Weitere Informationen oder Hilfe bei der Installation des DB2 Spatial Extender können Sie durch Anklicken von **Help** aufrufen.

Nach Abschluß der Installation ist DB2 Spatial Extender im Verzeichnis /usr/lpp/db2_07_01 installiert.

Installation überprüfen

Nach der Installation des DB2 Spatial Extender können Sie die Installation mit Hilfe des DB2 Spatial Extender-Beispielprogramms überprüfen. Bevor Sie das Beispielprogramm ausführen, müssen Sie die Datenbank SAMPLE erstellen und das Beispielprogramm ausführbar machen.

Anmerkung: Stellen Sie sicher, daß Sie den in der DB2 Spatial Extender Make-Datei angegebenen Compiler verwenden.

So kompilieren Sie das Beispielprogramm für Windows NT und führen es aus:

1. Melden Sie sich mit der Benutzer-ID an, die die Administratorberechtigung hat.
2. Geben Sie von einer Befehlszeile aus den Befehl **db2sampl** ein, um die DB2-Datenbank SAMPLE zu erstellen.
3. Geben Sie in einer Befehlszeile den folgenden Befehl ein:

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

Anmerkung: Zur Ausführung des Schritts 3 und zur weiteren Überprüfung der Installation müssen Sie über das Standard-DB2-Exemplar (DB2–DB2) verfügen.

4. Geben Sie **make rungsedemo** ein.
5. Geben Sie **rungsedemo.exe** ein.
6. Überprüfen Sie die bei der Ausführung des Programms angezeigten Fehler- und Abschlußnachrichten.

So kompilieren Sie das Beispielprogramm für AIX und führen es aus:

1. Melden Sie sich als Root an.
2. Erstellen oder aktualisieren Sie ein DB2-Exemplar.

3. Geben Sie von einer Befehlszeile aus den Befehl **db2sampl** ein, um die DB2-Datenbank SAMPLE zu erstellen.
4. Geben Sie in einer Befehlszeile den folgenden Befehl ein:

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

   Anmerkung: Zur Ausführung des Schritts 4 und zur weiteren Überprüfung der Installation müssen Sie über das erstellte bzw. aktualisierte Standard-DB2-Exemplar verfügen.
5. Geben Sie **make rungsedemo** ein.
6. Geben Sie **rungsedemo** ein.
7. Überprüfen Sie die bei der Ausführung des Programms angezeigten Fehler- und Abschlußnachrichten.

Informationen zu den Beispielprogrammen finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Überlegungen nach Abschluß der Installation

Nach Abschluß der Installation des DB2 Spatial Extender sollten Sie folgende Punkte in Betracht ziehen:

- ArcExplorer herunterladen
- Ausführen des Dienstprogramms zur Aktualisierung des DB2-Exemplars

Herunterladen von ArcExplorer

IBM verteilt ArcExplorer Java 3.0 als Beispielprogramm; Sie können dieses Programm aber auch von der ESRI Web-Site unter der Adresse http://www.esri.com abrufen.

Weitere Informationen zum Installieren und Verwenden von ArcExplorer finden Sie im Handbuch *Using ArcExplorer*, das Sie ebenfalls über die ESRI Web-Site abrufen können.

ArcExplorer erfordert die Java 2 Laufzeitumgebung (Standard Edition oder Enterprise Edition), V1.2.2; dieses Programm ist kostenlos über die Sun Web-Site unter der Adresse http://java.sun.com verfügbar.

Wichtig: DB2 Universal Database V7.1 wird zusammen mit IBM JDK 1.1.8 ausgeliefert. Wenn Sie JRE 1.2.2 für ArcExplorer installieren, verwenden Sie dafür ein anderes Verzeichnis als für DB2. Denken Sie daran, die Umgebungsvariable CLASSPATH entsprechend zu setzen.

Dienstprogramm zur Aktualisierung des DB2-Exemplars (db2iupdt) ausführen

Das Dienstprogramm db2iupdt aktualisiert ein angegebenes DB2-Exemplar wie folgt:

- Aktiviert das Exemplar für eine neue Systemkonfiguration.
- Aktiviert das Exemplar für den Zugriff auf eine Funktion, die der Installation oder dem Entfernen bestimmter Produktoptionen zugeordnet ist.

Unter AIX befindet sich dieses Dienstprogramm im Verzeichnis /usr/lpp/db2_07_01. Wenn Sie Hilfe benötigen, geben Sie `db2iupdt -h` in der Befehlszeile ein, um ein Hilfemenü zu öffnen. Im Betriebssystem Windows NT befindet sich db2iupdt im Verzeichnis \sqllib\bin. Wechseln Sie zur Eingabe dieses Befehls in dieses Verzeichnis. Eine umfassende Beschreibung dieses Befehls finden Sie im Handbuch *IBM DB2 Universal Database Command Reference*.

Wie geht es weiter?

Nach der Installation von DB2 Spatial Extender können Sie mit dem DB2 Control Center die GIS-Umgebung einrichten und die Arbeit mit räumlichen Informationen beginnen.

So rufen Sie den DB2 Spatial Extender über das Control Center auf:

1. Heben Sie im Fenster **Control Center** die Komprimierung des Objektbaums auf, bis Sie den Ordner **Databases** unter dem Server finden, auf dem DB2 Spatial Extender ausgeführt werden soll.
2. Klicken Sie den Ordner **Databases** an. Die Datenbanken werden im Inhalts-Teilfenster auf der rechten Seite des Fensters angezeigt.
3. Klicken Sie mit der rechten Maustaste die Datenbank an, mit der Sie arbeiten wollen, und klicken Sie die auszuführende räumliche Operation in dem Kontextmenü an.

Weitere Informationen zur Verwendung des DB2 Spatial Extender über das Control Center finden Sie in folgenden Kapiteln:

- „Kapitel 3. Ressourcen einrichten" auf Seite 25
- „Kapitel 4. Räumliche Spalten definieren, als Schichten registrieren und Geocodierer aktivieren" auf Seite 35
- „Kapitel 5. Räumliche Spalten ausfüllen" auf Seite 43
- „Kapitel 6. Räumliche Indizes erstellen" auf Seite 55

Kapitel 3. Ressourcen einrichten

Nach der Installation des DB2 Spatial Extender können Sie damit beginnen, in Ihre Datenbank Ressourcen einzutragen, die Sie beim Erstellen räumlicher Spalten und beim Bearbeiten räumlicher Daten benötigen. Dieses Kapitel faßt diese Ressourcen zusammen und beschreibt zwei der Aufgaben, über die Sie die Ressourcen verfügbar machen: Aktivieren der Datenbank für räumliche Operationen und Erstellen räumlicher Bezugssysteme.

Bestand der Ressourcen

Die Ressourcen, die Sie zum Erstellen räumlicher Spalten und zum Bearbeiten räumlicher Daten nutzen:

- *Bezugsdaten*: Adressen, die der DB2 Spatial Extender überprüft, um die Adressen für die Geocodierung zu überprüfen
- Ressourcen, die eine Datenbank für räumliche Operationen aktivieren: gespeicherte Prozeduren, räumliche Funktionen und andere
- Geocodierer, die von Benutzern und anderen Anbietern bereitgestellt werden (nicht der Standardcodierer)
- Räumliche Bezugssysteme

Dieser Abschnitt beschreibt Bezugsdaten und Ressourcen, die eine Datenbank für räumliche Operationen aktivieren. Informationen zu Geocodierern über den Standardcodierer hinaus finden Sie im Abschnitt „Informationen zur Geocodierung" auf Seite 43. Informationen zu räumlichen Bezugssystemen finden Sie im Abschnitt „Informationen zu Koordinaten und räumlichen Bezugssystemen" auf Seite 27.

Bezugsdaten

Bezugsdaten bestehen aus den neuesten Adressen in den USA, die vom United States Census Bureau erfaßt wurden. Bevor der Standard-Geocodierer eine Adresse in Ihrer Datenbank in Koordinaten umsetzen kann, muß er zunächst einen Teil oder alle Adressen mit den Adressen in den Bezugsdaten abgleichen.

Bezugsdaten werden verfügbar, wenn Sie den DB2 Spatial Extender installieren. Die Größe des für diese Daten erforderlichen Plattenspeicherplatzes ist im Abschnitt „Voraussetzungen für den Plattenspeicherplatz" auf Seite 20 beschrieben. Zur Überprüfung, ob die Daten unter AIX richtig geladen wurden, suchen Sie sie im Verzeichnis $DB2INSTANCE/sqllib/gse/refdata/. Zur Überprüfung, ob die Daten unter Windows NT richtig geladen wurden, suchen Sie sie im Verzeichnis %DB2PATH%\gse\refdata\.

Ressourcen, die eine Datenbank für räumliche Operationen aktivieren

Die erste Aufgabe, die Sie nach der Installation des DB2 Spatial Extender ausführen, ist das Aktivieren Ihrer Datenbank für räumliche Operationen. Hierzu gehört das Aufrufen einer Aktion, die bewirkt, daß der DB2 Spatial Extender die Datenbank mit den folgenden Ressourcen lädt:

- Gespeicherte Prozeduren. Wenn Sie eine Aktion über das Control Center anfordern, ruft der DB2 Spatial Extender zur Ausführung der Aktion eine dieser gespeicherten Prozeduren auf.
- Typen räumlicher Daten. Sie müssen jeder Tabelle oder Ansichtsspalte, in der räumliche Daten gespeichert werden sollen, einen Typ von räumlichen Daten zuordnen. Weitere Informationen hierzu finden Sie im Abschnitt „Informationen zu Typen von räumlichen Daten" auf Seite 36.
- Katalogtabellen und Sichten des DB2 Spatial Extender. Bestimmte Operationen hängen von dem DB2 Spatial Extender-Katalog ab. Bevor eine Spalte mit einem Typ von räumlichen Daten ausgefüllt werden kann, muß sie beispielsweise im Katalog als Schicht registriert werden. Informationen zu Schichten finden Sie im Abschnitt „Ein GIS-Projekt entwickeln und implementieren" auf Seite 12.
- Ein Typ eines räumlichen Index. Er ermöglicht Ihnen das Definieren von Indizes für Schichten.
- Räumliche Funktionen. Sie verwenden diese Funktionen zum Arbeiten mit räumlichen Daten auf verschiedene Arten, beispielsweise zum Ermitteln der Beziehung zwischen geographischen Merkmalen und zum Generieren weiterer räumlicher Daten. Eine dieser Funktionen ist ein Standard-Geocodierer. Er wandelt Adressen in den USA in Koordinaten um und fügt diese Koordinaten anschließend in räumliche Spalten ein. Weitere Informationen zu räumlichen Funktionen finden Sie in „Kapitel 13. Geometrien und zugeordnete räumliche Funktionen" auf Seite 129 und „Kapitel 14. Räumliche Funktionen für SQL-Abfragen" auf Seite 169. Weitere Informationen zum Standard-Geocodierer finden Sie im Abschnitt „Informationen zur Geocodierung" auf Seite 43.
- Ein Schema mit dem Namen DB2GSE, das die eben aufgelisteten Objekte enthält.

Anweisungen zur Verwendung des Control Center zum Initiieren des Aufrufs dieser Ressourcen finden Sie im Abschnitt „Eine Datenbank für räumliche Operationen aktivieren" auf Seite 27. Richtlinien zur Verwendung einer Routine in einem Anwendungsprogramm zur Ausführung der gleichen Aufgabe finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Eine Datenbank für räumliche Operationen aktivieren

Informationen darüber, welche Berechtigungen zum Aktivieren einer Datenbank für räumliche Operationen erforderlich sind, finden Sie im Abschnitt „Autorisierung" auf Seite 82.

So aktivieren Sie eine Datenbank für räumliche Operationen über das Control Center:

1. Heben Sie im Fenster **Control Center** die Komprimierung des Objektbaums auf, bis Sie den Ordner **Databases** unter dem Server finden, auf dem DB2 Spatial Extender ausgeführt werden soll.
2. Klicken Sie den Ordner **Databases** an. Die Datenbanken werden im Inhalts-Teilfenster auf der rechten Seite des Fensters angezeigt.
3. Klicken Sie mit der rechten Maustaste die Datenbank an, mit der Sie arbeiten wollen, und klicken Sie **Spatial Extender —> Enable** im Kontextmenü an. DB2 Spatial Extender stellt in der Datenbank die Ressourcen zur Verfügung, die Ihnen das Erstellen und Arbeiten mit räumlichen Spalten und Daten ermöglichen.

Erinnerung: Bevor Sie eine Datenbank für räumliche Operationen aktivieren können, muß DB2 Spatial Extender auf dem Server installiert sein, auf dem sich die Datenbank befindet.

Ein räumliches Bezugssystem erstellen

Dieser Abschnitt beschreibt die Beziehung zwischen räumlichen Bezugssystemen und Koordinatensystemen. Außerdem wird in diesem Abschnitt erläutert, wie ein räumliches Bezugssystem vom Control Center aus erstellt wird.

Informationen zu Koordinaten und räumlichen Bezugssystemen

Dieser Abschnitt setzt die im Abschnitt „Das Wesen der räumlichen Daten" auf Seite 6 begonnene Beschreibung von Koordinatensystemen fort. Anschließend wird die Definition räumlicher Bezugssysteme aus Abschnitt „Ein GIS-Projekt entwickeln und implementieren" auf Seite 12 erweitert. Der Abschnitt schließt mit Richtlinien für die Festlegung, welche Werte den Parametern eines räumlichen Bezugssystems zugeordnet werden sollen.

Koordinatensysteme, Koordinaten und Maße

Sie können sich ein Koordinatensystem wie ein imaginäres Gitter vorstellen, das einen bestimmten geographischen Bereich abdeckt. Beispiele hierfür sind ein Gitter über die gesamte Erde, über einen Staat oder über eine Region eines Landes. Jedes geographische Merkmal in diesem Bereich liegt im Schnittbereich einer Ost-West-Gitterlinie und einer Nord-Süd-Gitterlinie. Ein Wert, die *X-Koordinate*, gibt an, wo der Standort auf der Ost-West-Gitterlinie liegt. Ein anderer Wert, eine *Y-Koordinate*, gibt an, wo der Standort auf der Nord-

Süd-Gitterlinie liegt. Beide Werte stellen eine Beziehung des Standorts zum Mittelpunkt des Gitters, dem *Ursprung*, her.

Die X- und Y-Koordinaten sind am Ursprung beide Null. Vom Ursprung aus nach Osten sind die X-Koordinaten positiv, nach Westen hin sind sie negativ. Ebenso sind die Y-Koordinaten in nördlicher Richtung vom Ursprung aus positiv, in südlicher Richtung dagegen negativ. Eine Illustration dieser Verteilung bietet das folgende allgemeine Beispiel: Koordinatensystem A enthält ein Gitter, das ein großes Stadtgebiet abdeckt. Eine X-Koordinate von 7 kennzeichnet beispielsweise eine Position, die sieben Einheiten östlich vom Ursprung dieses Gitters liegt. Eine X-Koordinate -9.5 kennzeichnet zum Beispiel eine Position, die neun und eine halbe Einheit westlich vom Ursprung dieses Gitters liegt.

Jedes Datenelement in einer räumlichen Spalte enthält entweder (1) eine X-Koordinate und eine Y-Koordinate, die den Standort eines geographischen Merkmals definieren, oder (2) mehrere X- und Y-Koordinaten, die die Standorte von Teilen eines Merkmals definieren oder das von einem Merkmal abgedeckte Gebiet. Zwei weitere Arten von Werten — eine *Z-Koordinate* und ein *Maß*— können ebenfalls angegeben werden. Im Gegensatz zu X- und Y-Koordinaten werden Z-Koordinaten und Maß in DB2 Spatial Extender nicht zur Definition von Standorten oder Gebieten verwendet. Statt dessen enthalten sie Informationen, die von einer GIS-Anwendung benötigt werden. Eine Z-Koordinate gibt normalerweise die Höhe oder Tiefe eines geographischen Merkmals an. Z-Koordinaten über dem Ursprung sind positiv, darunter sind sie negativ. Ein Maß ist numerisch; es kann beliebige Arten von Informationen enthalten. Angenommen, Sie stellen in Ihrem GIS Ölquellen dar. Wenn Ihre Anwendungen Werte verarbeiten sollen, die Bohrstellen-IDs für seismische Daten kennzeichnen, können diese Daten als Maß gespeichert werden.

Räumliche Bezugssysteme, Abstände und Maßstabsfaktoren

Wie in „Koordinatensysteme, Koordinaten und Maße" auf Seite 27 beschrieben, können Koordinaten negativ sein und werden als Dezimalzahlen ausgedrückt. Dasselbe gilt für Maße. Zur Reduzierung des Speicheraufwands speichert DB2 Spatial Extender jede Koordinate und jedes Maß als nicht negative ganze Zahl (also als positive ganze Zahl oder als Null). Tatsächliche negative und dezimale Koordinaten und Maße müssen daher in nicht negativen ganzen Zahlen umgewandelt werden, damit der DB2 Spatial Extender sie speichern kann. Darüber hinaus müssen Sie DB2 Spatial Extender mitteilen, wie die Umwandlung erfolgen soll. Hierzu legen Sie bestimmte Parameter fest. Die zur Umwandlung von Koordinaten und Maßen innerhalb eines bestimmten geographischen Bereichs erforderlichen Parametereinstellungen werden zusammen als *räumliches Bezugssystem* bezeichnet.

Sie können ein räumliches Bezugssystem wie folgt erstellen:

- Durch Ermitteln der niedrigsten negativen Koordinaten und Maße für die dargestellten Merkmale. (Je weiter ein negativer Wert von Null entfernt ist, desto niedriger ist er. Eine X-Koordinate von –10 ist niedriger als eine X-Koordinate von –5; ein Maß von –100 ist niedriger als ein Maß von –50.)
- Durch Angeben von *Abstandsfaktoren* (kurz: *Abständen*): Werte, die von negativen Koordinaten und Maßen subtrahiert werden und somit nicht negative Zahlen ergeben.
- Durch Angeben von *Maßstabsfaktoren*: Werte, die durch Multiplikation mit dezimalen Koordinaten und Maßen ganze Zahlen ergeben, deren Genauigkeit mindestens der Genauigkeit der Koordinaten oder Maße entspricht. Betrachten Sie beispielsweise eine Koordinate mit der Genauigkeit vier: 92.77. Sie können diese Koordinate mit dem Maßstabsfaktor 100 multiplizieren, um eine ganze Zahl mit der Genauigkeit vier zu erhalten: 9277.

Die niedrigsten negativen Koordinaten und Maße ermitteln

Bevor Sie die Parameter für ein räumliches Bezugssystem festlegen, müssen Sie die niedrigste negative X-, Y- und Z-Koordinate sowie das Maß in dem geographischen Bereich, der die entsprechenden Merkmale enthält, ermitteln. Sie können diese Werte durch Beantworten der folgenden Fragen ermitteln:

- Liegen irgendwelche der dargestellten Merkmale westlich vom Ursprung des verwendeten Koordinatensystems? Falls ja, welche X-Koordinaten geben die Position des westlichen Rands des westlichsten Merkmals an? (Die Antwort gibt die niedrigste negative X-Koordinate an, die Sie hier behandeln.) Wenn Sie beispielsweise Ölquellen darstellen und einige dieser Quellen westlich des Ursprungs liegen, welche X-Koordinate gibt die Position der westlichsten Ölquelle an?
- Liegen irgendwelche der Merkmale südlich des Ursprungs? Falls ja, welche Y-Koordinaten geben die Position des südlichsten Rands des südlichsten Merkmals an? (Die Antwort gibt die niedrigste negative Y-Koordinate an, die Sie hier behandeln.) Wenn Sie beispielsweise Ölquellen darstellen und einige dieser Quellen südlich des Ursprungs liegen, welche X-Koordinate gibt die Position der südlichsten Ölquelle an?
- Wenn Sie die Tiefe mit Z-Koordinaten definieren wollen, welches Merkmal ist das tiefste, und welche Z-Koordinate kennzeichnet den tiefsten Punkt dieses Merkmals? (Die Antwort gibt die niedrigste negative Z-Koordinate an, die Sie hier behandeln.)
- Wenn Sie Maße in Ihre räumlichen Daten einbeziehen wollen, können dabei auch negative Maße vorkommen? Falls ja, welches ist der niedrigste Wert für ein negatives Maß?

Nachdem Sie die niedrigsten negativen Koordinaten bestätigt haben, addieren Sie dazu einen Wert von fünf bis zehn Prozent. Wenn die niedrigste negative X-Koordinate beispielsweise –100 ist, könnten Sie –5 addieren. Die so erhaltene Zahl wird in diesem Handbuch als *erweiterter Wert* bezeichnet.

Abstandsfaktoren angeben

Geben Sie als nächstes an, welche Abstandsfaktoren der DB2 Spatial Extender zur Umwandlung negativer Koordinaten und Maße in nicht negative Zahlen verwenden soll:

- Nachdem Sie festgelegt haben, was Ihr erweiterter X-Wert sein soll, geben Sie einen Abstand an, der nach Subtraktion von diesem Wert Null ergibt. Der DB2 Spatial Extender subtrahiert diese Zahl von allen negativen X-Koordinaten, um einen positiven Wert zu erhalten. Der DB2 Spatial Extender subtrahiert diese Zahl auch von allen anderen X-Koordinaten.

  Wenn der erweiterte X-Wert beispielsweise –105 lautet, müssen Sie –105 davon subtrahieren, um Null zu erhalten. Der DB2 Spatial Extender subtrahiert dann –105 von allen X-Koordinaten, die den dargestellten Merkmalen zugeordnet sind. Da keine dieser Koordinaten größer als –100 ist, sind alle durch diese Subtraktion erhaltenen Werte positiv.
- Geben Sie in der gleichen Weise Abstandswerte an, die nach der Subtraktion von dem erweiterten Y-Wert, dem erweiterten Z-Wert oder dem erweiterten Maß Null ergeben.

Der von den X-Koordinaten subtrahierte Abstand wird als *falsches X* bezeichnet. Die von den Y-Koordinaten, den Z-Koordinaten und dem Maß zu subtrahierenden Abstände werden entsprechend als *falsches Y*, *falsches Z* und *falsches M* bezeichnet. Anweisungen zum Angeben dieser Parameter über das Control Center finden Sie im Abschnitt „Über das Control Center ein räumliches Bezugssystem erstellen" auf Seite 31.

Maßstabsfaktoren angeben

Geben Sie als nächstes an, welche Maßstabsfaktoren der DB2 Spatial Extender zur Umwandlung dezimaler Koordinaten und Maße in ganze Zahlen verwenden soll:

- Geben Sie einen Maßstabsfaktor an, der bei Multiplikation mit einer dezimalen X-Koordinate oder einer dezimalen Y-Koordinate eine ganze 32-Bit-Zahl ergibt. Es empfiehlt sich, für diesen Maßstabsfaktor eine Potenz von zehn zu verwenden: 10 hoch eins (10), 10 hoch zwei (100), 10 hoch drei (1000) oder ggf. eine höhere Zehnerpotenz. Legen Sie die Zehnerpotenz wie folgt fest:
  1. Ermitteln Sie, welche X- und Y-Koordinaten Dezimalzahlen sind bzw. sein werden. Angenommen, Sie stellen fest, daß drei der verschiedenen X- und Y-Koordinaten, mit denen Sie arbeiten, Dezimalzahlen sind: 1.23, 5.1235 und 6.789.
  2. Verwenden Sie die Dezimalkoordinate mit der längsten Dezimalgenauigkeit. Stellen Sie anschließend fest, mit welcher Zehnerpotenz diese Koordinate multipliziert werden kann, um eine ganze Zahl mit entsprechender Genauigkeit zu erhalten. In unserem Beispiel hat 5.1235

von den drei Dezimalkoordinaten die längste Dezimalgenauigkeit. Durch eine Multiplikation mit zehn hoch vier (10000) ergibt sich die Dezimalzahl 51235.

3. Stellen Sie fest, ob die durch die Multiplikation erhaltene ganze Zahl zu lang ist, um in sie in einem 32-Bit-Datenelement zu speichern. 51235 ist nicht zu lang. Aber nehmen Sie an, neben 1.23, 5.11235 und 6.789 umfaßt der Bereich Ihrer X- und Y-Koordinaten noch einen vierten Dezimalwert 10006.789876. Die Dezimalgenauigkeit dieser Koordinate ist länger als die der anderen drei Dezimalkoordinaten; Sie multiplizieren daher *diese* Koordinate — nicht 5.1235 — mit dem Faktor 10. Zur Umwandlung in eine ganze Zahl könnten Sie sie mit zehn hoch sechs (1000000) multiplizieren. Der dadurch erhaltene Wert 10006789876 ist jedoch zu lang, um ihn in einem 32-Bit-Datenelement speichern zu können. Wenn der DB2 Spatial Extender versucht, diesen Wert zu speichern, sind die Ergebnisse nicht vorhersehbar.

   Vermeiden Sie dieses Problem, indem Sie eine Potenz von zehn wählen, die bei der Multiplikation mit der ursprünglichen Koordinate eine Dezimalzahl ergibt, die der DB2 Spatial Extender so abschneiden kann, daß sich bei minimalem Genauigkeitsverlust eine ganze Zahl ergibt, die gespeichert werden kann. In diesem Fall können Sie zehn hoch vier (10000) wählen. Die Multiplikation von 10000 mit 10006.789876 ergibt 100067898.76. Der DB2 Spatial Extender schneidet diese Nummer auf 100067898 ab und reduziert damit ihre Genauigkeit um eine praktisch unbedeutende Größe.

- Wenn die dargestellten Merkmale dezimale Z-Koordinaten enthalten, ermitteln Sie anhand der vorangegangenen Prozedur einen geeigneten Maßstabsfaktor für diese Koordinaten. Wenn die Merkmale dezimalen Maßen zugeordnet sind, verwenden Sie diese Prozedur auch zum Ermitteln eines Maßstabsfaktors für diese Maße.

Der Maßstabsfaktor für X- und Y-Koordinaten wird als *XY-Einheit* bezeichnet. Die Skalierungsfaktoren für Z-Koordinaten und Maße werden als *Z-Einheiten* bzw. *M-Einheiten* bezeichnet. Anweisungen zum Angeben dieser Parameter über das Control Center finden Sie im Abschnitt „Über das Control Center ein räumliches Bezugssystem erstellen".

Über das Control Center ein räumliches Bezugssystem erstellen

Dieser Abschnitt enthält eine Übersicht über die erforderlichen Schritte zum Erstellen eines räumlichen Bezugssystems über das Control Center. Im Anschluß an diese Übersicht finden Sie Details zu Durchführung der einzelnen Schritte.

Zur Ausführung dieser Schritte ist keine spezielle Berechtigung erforderlich.

Übersicht über die erforderlichen Schritte zum Erstellen eines räumlichen Bezugssystems über das Control Center:

1. Öffnen Sie das Fenster **Create Spatial Reference**.
2. Geben Sie an, welches Koordinatensystem verwendet werden soll.
3. Geben Sie die Kennungen für das zu erstellende räumliche Bezugssystem an.
4. Stellen Sie fest, welche Bereiche von Koordinaten und Maßen für die geographischen Merkmale gelten, zu denen Sie Informationen erhalten wollen.
5. Geben Sie Werte an für die Umwandlung negativer oder dezimaler Koordinaten und Maße in Datenelemente, die der DB2 Spatial Extender speichern kann.
6. Weisen Sie den DB2 Spatial Extender an, das gewünschte räumliche Bezugssystem zu erstellen.

Ausführliche Schritte zum Erstellen eines räumlichen Bezugssystems über das Control Center:

1. Öffnen Sie das Fenster **Create Spatial Reference**.
   a. Heben Sie im Fenster **Control Center** die Komprimierung des Objektbaums auf, bis Sie den Ordner **Databases** unter dem Server finden, auf dem der DB2 Spatial Extender ausgeführt werden soll.
   b. Klicken Sie den Ordner **Databases** an. Die Datenbanken werden im Inhalts-Teilfenster auf der rechten Seite des Fensters angezeigt.
   c. Klicken Sie mit der rechten Maustaste die Datenbank an, die Sie für die räumlichen Daten aktiviert haben. Klicken Sie anschließend **Spatial Extender —> Spatial References** im Kontextmenü an. Das Fenster "Spatial References" wird geöffnet.
   d. Klicken Sie im Fenster "Spatial References" **Create** an. Das Fenster **Create Spatial Reference** wird geöffnet.
2. Verwenden Sie im Fenster **Create Spatial Reference** das Feld **Coordinate system**, um anzugeben, welches Koordinatensystem verwendet werden soll.
3. Geben Sie die Kennungen für das zu erstellende räumliche Bezugssystem an.
   - Geben Sie im Feld **Name** einen Namen aus 1 - 64 Zeichen für das System ein.

     Einschränkung: Geben Sie nicht den Namen eines anderen räumlichen Bezugssystems an. Zwei räumliche Bezugssysteme in der Datenbank können nicht den gleichen Namen haben.

- Geben Sie im ID-Feld eine numerische Kennung an. Diese Kennung muß eine ganze Zahl sein.

  Einschränkung: Geben Sie nicht die ID eines anderen räumlichen Bezugssystems an. Zwei räumliche Bezugssysteme in der Datenbank können nicht die gleiche ID haben.

4. Ermitteln Sie mit einem Medium außerhalb des Control Center — beispielsweise auf Papier oder einer weißen Tafel — die niedrigsten negativen Koordinaten und Maße für die dargestellten geographischen Merkmale. Anweisungen hierzu finden Sie im Abschnitt „Die niedrigsten negativen Koordinaten und Maße ermitteln" auf Seite 29.
5. Geben Sie im Fenster **Create Spatial Reference** Werte an für die Umwandlung negativer oder dezimaler Koordinaten und Maße in Datenelemente, die vom DB2 Spatial Extender unterstützt werden — also nicht-negative ganze 32-Bit-Zahlen.
   a. Geben Sie Werte an für die Umwandlung negativer oder dezimaler X-Koordinaten in nicht-negative ganze Zahlen:
      - Geben Sie in der Spalte **Offset** in dem Feld neben **X** ein "falsches X" ein:
        – Wenn irgendwelche Werte im Bereich der in Schritt 4 angegebenen X-Koordinaten negativ sind, geben Sie ein "falsches X" ein, das bei der Subtraktion von der niedrigsten negativen Koordinate eine positive Zahl ergibt. Anweisungen hierzu finden Sie im Abschnitt „Abstandsfaktoren angeben" auf Seite 30.
        – Wenn keine der X-Koordinaten negativ ist, geben Sie ein falsches X von 0 ein.
      - Geben Sie in der Spalte **Scale factor** eine XY-Einheit in dem Feld ganz rechts von **X** ein. Diese XY-Einheit sollte bei Multiplikation mit einer beliebigen dezimalen X- oder Y-Koordinate eine ganze Zahl ergeben, die mit minimalem Genauigkeitsverlust als 32-Bit-Datenelement gespeichert werden kann. Anweisungen hierzu finden Sie im Abschnitt „Maßstabsfaktoren angeben" auf Seite 30.

        Nach der Angabe der XY-Einheit in dem Feld ganz rechts von **X** erscheint diese Einheit auch in dem Feld ganz rechts von **Y**.

   b. Geben Sie ein "falsches Y" ein, das dem DB2 Spatial Extender die Umwandlung negativer Y-Koordinaten in positive Werte ermöglicht. Geben Sie diesen Wert in der Spalte **Offset** in dem Feld neben **Y** ein:
      - Wenn irgendwelche Werte im Bereich der in Schritt 4 angegebenen Y-Koordinaten negativ sind, geben Sie ein "falsches Y" ein, das bei der Subtraktion von der niedrigsten negativen Koordinate eine positive Zahl ergibt. Anweisungen hierzu finden Sie im Abschnitt „Abstandsfaktoren angeben" auf Seite 30.
      - Wenn alle Y-Koordinaten positiv sind, geben Sie ein "falsches Y" von 0 ein.

c. Wenn Sie Z-Koordinaten in Ihre räumlichen Daten einbeziehen wollen, geben Sie Werte an zur Umwandlung negativer oder dezimaler Z-Koordinaten in nicht-negative ganze Zahlen.
   - Geben Sie in der Spalte **Offset** in dem Feld neben **Z** ein "falsches Z" ein:
     – Wenn irgendwelche Werte im Bereich der in Schritt 4 auf Seite 33 angegebenen Z-Koordinaten negativ sind, geben Sie ein "falsches Z" ein, das bei der Subtraktion von der niedrigsten negativen Koordinate eine positive Zahl ergibt. Anweisungen hierzu finden Sie im Abschnitt „Abstandsfaktoren angeben" auf Seite 30.
     – Wenn keine der Z-Koordinaten negativ ist, geben Sie ein falsches Z von 0 ein.
   - Geben Sie in der Spalte **Scale factor** eine Z-Einheit in dem Feld ganz rechts von **Z** ein. Diese Z-Einheit sollte bei Multiplikation mit einer beliebigen dezimalen Z-Koordinate eine ganze Zahl ergeben, die mit minimalem Genauigkeitsverlust als 32-Bit-Datenelement gespeichert werden kann. Anweisungen hierzu finden Sie im Abschnitt „Maßstabsfaktoren angeben" auf Seite 30.

d. Wenn Sie Maße in Ihre räumlichen Daten einbeziehen wollen, geben Sie Werte an zur Umwandlung negativer oder dezimaler Maße in nicht-negative ganze Zahlen.
   - Geben Sie in der Spalte **Offset** in dem Feld neben **Linear** ein "falsches M" ein:
     – Wenn irgendwelche Werte im Bereich der in Schritt 4 auf Seite 33 angegebenen Maße negativ sind, geben Sie ein "falsches M" ein, das bei der Subtraktion von dem niedrigsten negativen Maß eine positive Zahl ergibt. Anweisungen hierzu finden Sie im Abschnitt „Abstandsfaktoren angeben" auf Seite 30.
     – Wenn alle Maße positiv sind, geben Sie ein "falsches M" von 0 ein.
   - Geben Sie in der Spalte **Scale factor** eine M-Einheit in dem Feld ganz rechts von der Bezeichnung **Linear** ein. Diese M-Einheit sollte bei Multiplikation mit einem beliebigen Maß eine ganze Zahl ergeben, die mit minimalem Genauigkeitsverlust als 32-Bit-Datenelement gespeichert werden kann. Anweisungen hierzu finden Sie im Abschnitt „Maßstabsfaktoren angeben" auf Seite 30.

6. Klicken Sie **OK** an, um das gewünschte räumliche Bezugssystem zu erstellen.

Kapitel 4. Räumliche Spalten definieren, als Schichten registrieren und Geocodierer aktivieren

Nachdem Sie Ressourcen für Ihr DB2 Spatial Extender-GIS eingerichtet haben, können Sie Objekte erstellen, die räumliche Daten enthalten. Wenn Sie beispielsweise neue Tabellen benötigen, die räumliche Daten enthalten sollen, können Sie diese definieren, indem Sie den Spalten Typen von räumlichen Daten zuordnen. Oder Sie können vorhandenen Tabellen räumliche Spalten hinzufügen.

Wenn Sie in einer neuen oder einer vorhandenen Tabelle eine räumliche Spalte definieren, müssen Sie diese Spalte als Schicht registrieren. Darüber hinaus können Sie, wenn die Spalte über einen Geocodierer ausgefüllt werden soll, bei der Registrierung der Spalte als Schicht den Geocodierer aktivieren, damit er die Spalte automatisch verwaltet. Diese Aktivierung erfolgt auf die folgende Weise: Der DB2 Spatial Extender definiert Auslöser, die den Geocodierer immer aufrufen, wenn die entsprechende attributive Spalte (bzw. Spalten) der räumlichen Spalte neue oder aktualisierte Daten empfängt. Beim Aufruf wandelt der Geocodierer die neuen oder aktualisierten Daten in räumliche Daten um und legt diese Daten in der räumlichen Spalte ab.

Nach der Definition einer räumlichen Spalte für die Tabelle können Sie bei Bedarf eine Sichtspalte über dieser Tabellenspalte erstellen. Sie müssen die Sichtspalte als Schicht registrieren, nachdem Sie die Tabellenspalte als Schicht registriert haben.

In diesem Kapitel wird das Wesen und die Verwendung der Datentypen, die Sie der räumlichen Spalte zuordnen können, beschrieben. Als nächstes beschreibt das Kapitel, wie mit dem Control Center eine räumliche Spalte für eine Tabelle definiert wird, um diese Spalte als Spalte zu registrieren, und wie ein Geocodierer für die Verwaltung der Schicht aktiviert wird. Das Kapitel schließt mit einer Erläuterung, wie über das Control Center eine Sichtspalte als Schicht registriert wird.

Informationen zu Typen von räumlichen Daten

Dieser Abschnitt stellt die für räumliche Spalten erforderlichen Datentypen vor und bietet Richtlinien zur Auswahl eines geeigneten Datentyps für eine räumliche Spalte.

Wenn Sie eine Datenbank für räumliche Operationen aktivieren, stellt der DB2 Spatial Extender der Datenbank eine Hierarchie strukturierter Datentypen zur Verfügung. Abb. 6 zeigt diese Hierarchie. In dieser Abbildung haben die Exemplartypen einen weißen Hintergrund; die nicht exemplarfähigen Typen sind auf einem schraffierten Hintergrund dargestellt.

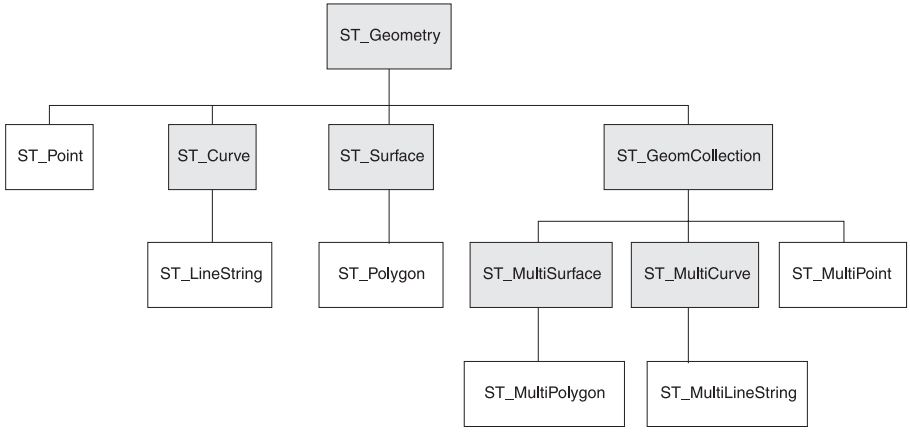


Abbildung 6. Hierarchie der Typen räumlicher Daten. Die Datentypen in weißen Kästchen sind exemplarfähig. Datentypen in schraffierten Kästchen sind nicht exemplarfähig.

Die Hierarchie in Abb. 6 umfaßt:

- Datentypen für geographische Merkmale, die eine Einheit bilden, beispielsweise einzelne Wohngebiete und isolierte Seen.
- Datentypen für geographische Merkmale, die aus mehreren Einheiten oder Komponenten bestehen, beispielsweise Fernstraßensysteme und Gebirgszüge.
- Ein Datentyp für geographische Merkmale aller Art.

Datentypen für Einheitenmerkmale

Verwenden Sie ST_Point, ST_Linestring und ST_Polygon zum Speichern von Koordinaten zur Definition des Raumes, der von Merkmalen belegt wird, die eine einzelne Einheit bilden:

- Verwenden Sie ST_Point, wenn Sie einen Punkt im Raum kennzeichnen wollen, der von einem eindeutigen geographischen Merkmal belegt wird. Das Merkmal kann ein sehr kleines sein, beispielsweise eine Wasserquelle, ein großes wie beispielsweise eine Stadt, oder ein mittleres wie ein Gebäudekomplex oder ein Park. In jedem Fall kann der Punkt im Raum am Kreuzungspunkt der Ost-West-Koordinatenlinie (z. B. einem Breitengrad) mit einer Nord-Süd-Koordinatenlinie (z. B. einem Längengrad) gefunden werden. Ein ST_Point-Datenelement enthält Werte — eine X-Koordinate und eine Y-Koordinate — zur Definition eines solchen Kreuzungspunkts. Die X-Koordinate gibt an, wo der Punkt auf der Ost-West-Linie liegt; die Y-Koordinate gibt an, wo er auf der Nord-Süd-Linie liegt.
- Verwenden Sie ST_Linestring für Koordinaten, die den Raum definieren, der von linearen Funktionen belegt wird, beispielsweise Straßen, Kanäle oder Rohrleitungen.
- Verwenden Sie ST_Polygon, wenn Sie einen Bereich kennzeichnen wollen, der von einem mehrseitigen Merkmal abgedeckt wird, beispielsweise ein Industriegebiet, einen Wald oder ein Naturschutzgebiet. Ein ST_Polygon-Datenelement besteht aus den Koordinaten, die den Umfang eines solchen Merkmals definieren.

  In manchen Fällen können ST_Polygon und ST_Point für dieselbe Funktion verwendet werden. Angenommen, Sie benötigen räumliche Informationen über verschiedene Apartmentkomplexe. Wenn Sie einen Punkt im Raum darstellen wollen, an dem sich die einzelnen Komplexe befinden, verwenden Sie ST_Point zum Speichern der X- und Y-Koordinaten, die einen solchen Punkt definieren. Wenn Sie andererseits den Bereich darstellen wollen, den die einzelnen Komplexe abdecken, verwenden Sie ST_Polygon zum Speichern der Koordinaten, die den Umfang des Bereichs angeben.

Datentypen für Merkmale mit mehreren Einheiten

Verwenden Sie ST_MultiPoint, ST_MultiLineString und ST_MultiPolygon zum Speichern von Koordinaten für den Raum, der von Merkmalen belegt wird, die aus mehreren Einheiten bestehen:

- Verwenden Sie ST_MultiPoint, wenn Sie Merkmale darstellen wollen, die aus eindeutigen Einheiten bestehen, und wenn Sie außerdem den Punkt im Raum angeben wollen, der von den einzelnen Komponenten belegt wird. Ein ST_MultiPoint-Datenelement enthält die Paare der X- und Y-Koordinaten, die die Position jeder Komponente eines solchen Merkmals definieren. Betrachten Sie beispielsweise eine Tabelle, deren Zeilen Inselketten darstellen und deren Spalten eine ST_MultiPoint-Spalte enthalten. Jedes Datenelement in der Spalte enthält die Paare der X- und Y-Koordinaten, die die Position der Inseln in den einzelnen Ketten definieren.
- Verwenden Sie ST_MultiLineString, wenn Sie Merkmale darstellen wollen, die aus linearen Einheiten bestehen, und wenn Sie außerdem Informationen zu dem von jeder einzelnen Einheit belegten Raum abfragen wollen. Ein ST_MultiLineString-Datenelement besteht aus den Koordinaten, die solche Räume definieren. Betrachten Sie beispielsweise eine Tabelle, deren Zeilen Flußsysteme darstellen und deren Spalten eine ST_MultiLineString-Spalte enthalten. Jedes Datenelement in dieser Spalte enthält die Koordinatengruppen, die den Verlauf der Flüsse in den einzelnen Systemen definieren.
- Verwenden Sie ST_MultiPolygon, wenn Sie Merkmale darstellen wollen, die aus mehrseitigen Einheiten bestehen, und wenn Sie außerdem Informationen zu dem von jeder einzelnen Einheit belegten Raum abfragen wollen. Betrachten Sie beispielsweise eine Tabelle, deren Zeilen Landkreise im Norden Deutschlands darstellen und deren Spalten eine ST_MultiPolygon-Spalte enthalten. Diese Spalte enthält Informationen zu Bauernhöfen. Jedes Datenelement in der Spalte enthält die Koordinatengruppen, die den Umfang der Bauernhöfe in einem bestimmten Landkreis definieren.

Ein Datentyp für alle Merkmale

Sie können ST_Geometry verwenden, wenn Sie nicht sicher sind, welcher der anderen Datentypen geeignet ist. Da ST_Geometry der Ausgangspunkt der Hierarchie ist, zu dem die anderen Datentypen gehören, kann eine ST_Geometry-Spalte einige oder alle Werte speichern, die in Spalten gespeichert werden können, denen andere Datentypen zugeordnet sind.

Achtung: Der Standard-Geocodierer kann Adressen in ST_Point- oder ST_Geometry-Datenelemente umwandeln. Wenn Sie daher vorhaben, mit diesem Geocodierer eine räumliche Spalte auszufüllen, müssen Sie dieser Spalten einen der Datentypen ST_Point oder ST_Geometry zuordnen.

Eine räumliche Spalte für eine Tabelle definieren, diese Spalte als Schicht registrieren und einen Geocodierer zum Verwalten aktivieren

Dieser Abschnitt gibt einen Überblick über die erforderlichen Schritte zum Definieren einer räumlichen Spalte für eine Tabelle, zum Registrieren dieser Spalte als Schicht und zum Aktivieren eines Geocodierers für die Verwaltung. Im Anschluß an diese Übersicht finden Sie Details zu Durchführung der einzelnen Schritte.

Informationen darüber, welche Berechtigung Sie zum Registrieren einer Tabellenspalte als Schicht benötigen, finden Sie im Abschnitt „Autorisierung" auf Seite 96. Informationen darüber, welche Berechtigung Sie zum Aktivieren eines Geocodierers zum Verwalten einer Tabellenspalte als Schicht benötigen, finden Sie im Abschnitt „Autorisierung" auf Seite 78.

Überblick über die Schritte zum Definieren einer räumlichen Spalte als Tabelle, zum Registrieren dieser Spalte als Schicht und zum Aktivieren eines Geocodierers für die Verwaltung:

1. Wenn die räumliche Spalte Teil einer neuen Tabelle sein soll, erstellen Sie diese Tabelle.
2. Öffnen Sie das Fenster **Create Spatial Layer**.
3. Fügen Sie entweder eine räumliche Spalte einer Tabelle hinzu und geben Sie an, daß diese Spalte als Schicht registriert werden soll, oder geben Sie an, daß eine vorhandene Spalte als Schicht registriert werden soll.
4. Geben Sie an, welches räumliche Bezugssystem für die Schicht verwendet werden soll.
5. Wenn die Schicht importierte Daten enthält oder die Daten aus einer anderen räumlichen Spalte generiert werden, weisen Sie den DB2 Spatial Extender an, die Schicht zu erstellen.
6. Wenn die Schicht Daten enthalten soll, die aus attributiven Daten abgeleitet wurden:
   a. Geben Sie an, welche Spalte bzw. Spalten diese attributiven Daten enthält.
   b. Geben Sie an, daß Sie einen Geocodierer zur Verwaltung der Schicht aktivieren wollen.
   c. Weisen Sie den DB2 Spatial Extender an, die Schicht zu erstellen.

Ausführliche Informationen zu den Schritten zum Definieren einer räumlichen Spalte als Tabelle, zum Registrieren dieser Spalte als Schicht und zum Aktivieren eines Geocodierers für die Verwaltung:

1. Wenn die räumliche Spalte Teil einer neuen Tabelle sein soll, erstellen Sie diese Tabelle:
   - Verwenden Sie eine Schnittstelle Ihrer Wahl (z. B. das Control Center oder den Befehlszeilenprozessor) zum Erstellen der Tabelle.
   - Wenn Sie einen Geocodierer verwenden wollen, beziehen Sie zwischen einer und zehn Spalten ein, mit denen der Geocodierer arbeiten kann. Ein Geocodierer kann nicht mehr als zehn Spalten als Eingabe verarbeiten.
   - Beziehen Sie entweder die räumliche Spalte, die Sie als Schicht registrieren wollen, ein, oder definieren Sie diese Spalte in Schritt 3.

   Wenn Sie eine vorhandene Tabelle verwenden wollen fahren Sie mit dem nächsten Schritt fort.
2. Öffnen Sie das Fenster **Create Spatial Layer**.
   a. Heben Sie im Fenster **Control Center** die Komprimierung des Objektbaums auf, bis Sie den Ordner **Tables** für die Tabellen in der Datenbank finden, die Sie für die räumlichen Operationen verwenden wollen.
   b. Klicken Sie den Ordner **Tables** an. Die Tabellen werden im Inhalts-Teilfenster auf der rechten Seite des Fensters angezeigt.
   c. Klicken Sie mit der rechten Maustaste die gewünschte Tabelle an, und klicken Sie **Spatial Extender —> Spatial Layers** im Kontextmenü an. Das Fenster **Spatial Layers** wird geöffnet.
   d. Klicken Sie im Fenster **Spatial Layers** die Option **Create** an. Das Fenster **Create Spatial Layer** wird geöffnet.
3. Fügen Sie im Fenster **Create Spatial Layer** entweder eine räumliche Spalte einer Tabelle hinzu und geben Sie an, daß diese Spalte als Schicht registriert werden soll, oder geben Sie an, daß eine vorhandene Spalte als Schicht registriert werden soll.
   - So fügen Sie der Tabelle eine räumliche Spalte hinzu und definieren diese Spalte als Schicht:
     a. Geben Sie im Feld **Layer column** einen Namen für die Spalte ein.
     b. Wählen Sie im Feld **Column type** den gewünschten Datentyp für die Spalte aus bzw. geben Sie ihn ein. Eine Beschreibung der zulässigen Datentypen finden Sie im Abschnitt „Informationen zu Typen von räumlichen Daten" auf Seite 36.
   - Wenn Sie vorhandene Spalte als Schicht definieren wollen, wählen Sie sie im Feld **Layer column** aus.

     Einschränkung: Wählen Sie keine Spalte aus, die bereits als Schicht definiert wurde.

4. Geben Sie im Feld **Spatial reference name** den Namen des räumlichen Bezugssystems an, der für die Schicht verwendet werden soll.
5. Wenn die Schicht importierte Daten enthalten soll oder Daten, die aus einer anderen räumlichen Spalte generiert wurden, klicken Sie **OK** an, um sie zu registrieren.
6. Wenn die Schicht Daten enthalten soll, die aus attributiven Daten abgeleitet wurden, gehen Sie wie folgt vor:
   a. Geben Sie an, welche Spalte bzw. Spalten diese attributiven Daten enthält:
      1) Wählen Sie die Spalte(n) im Feld **Available columns** aus. Sie können bis zu zehn Spalten auswählen.
      2) Klicken Sie den Druckknopf >, den Druckknopf >> oder beide an, um die ausgewählte Spalten im Feld **Selected columns** aufzulisten.
   b. So aktivieren Sie einen Geocodierer zur Verwaltung der Schicht:
      1) Wählen Sie das Markierungsfeld **Enable automatic geocoder** aus.
      2) Wählen Sie im Feld **Name** den Namen des Geocodierers aus, den Sie verwenden wollen.
      3) Geben Sie im Feld **Precision level** den Grad in Prozent an, in dem die Eingabedatensätze mit den entsprechenden Datensätzen in den Bezugsdaten übereinstimmen müssen, damit sie verarbeitet werden können. Dieser Prozentsatz wird als *Genauigkeit* bezeichnet. Angenommen, der Geocodierer liest einen Eingabedatensatz mit der Adresse 557 Bailey, San Jose 94120, ein. Wenn die Genauigkeit 100 ist und die Übereinstimmung zwischen dieser Adresse und ihrem Pendant in den Bezugsdaten nicht 100 Prozent beträgt, weist der Geocodierer die Adresse zurück. Ist die Genauigkeit 75 und die Übereinstimmung zwischen dem Datensatz und dem Pendant in den Bezugsdaten beträgt mindestens 75 Prozent, verarbeitet der Geocodierer die Adresse.
      4) Wenn der Geocodierer von einem separaten Anbieter bereitgestellt wurde, geben Sie im Feld **Properties** alle herstellerspezifischen Geocodierungsparameter an, die Sie verwenden wollen.
   c. Klicken Sie **OK** an, um die ausgewählte Spalte als Schicht zu registrieren und, falls Sie diese Funktion angefordert haben, zum Aktivieren des Geocodierers für die Verwaltung der Spalte.

Eine Sichtspalte als Schicht registrieren

Informationen darüber, welche Berechtigung Sie zum Registrieren einer Sichtspalte als Schicht benötigen, finden Sie im Abschnitt „Autorisierung” auf Seite 96.

So registrieren Sie eine Sichtspalte als Schicht:

1. Öffnen Sie das Fenster **Create Spatial Layer**.
   a. Heben Sie im Fenster **Control Center** die Komprimierung des Objektbaums auf, bis Sie den Ordner **Views** für die Sichten in der Datenbank finden, die Sie für die räumlichen Operationen verwenden wollen.
   b. Klicken Sie den Ordner **Views** an. Die Sichten werden im Inhalts-Teilfenster auf der rechten Seite des Fensters angezeigt.
   c. Klicken Sie mit der rechten Maustaste die gewünschte Sicht an, und klicken Sie **Spatial Extender —> Spatial Layers** im Kontextmenü an. Das Fenster **Spatial Layers** wird geöffnet.
   d. Klicken Sie im Fenster **Spatial Layers** die Option **Create** an. Das Fenster **Create Spatial Layer** wird geöffnet.
2. Geben Sie im Feld **Layer column** die Spalte an, die Sie als Schicht registrieren wollen.
3. Geben Sie im Feld **Underlying spatial layer** den Namen der Tabellenspalte ein, auf der die ausgewählte Sichtspalte basiert. Diese Tabellenspalte muß bereits als Schicht registriert worden sein.
4. Klicken Sie **OK** an, um die angegebene Sichtspalte als Schicht zu registrieren.

Kapitel 5. Räumliche Spalten ausfüllen

Nachdem Sie räumliche Spalten als Schichten registriert haben, können Sie sie mit räumlichen Daten ausfüllen. Wie im Abschnitt „Woher räumliche Daten stammen" auf Seite 7 angegeben, können diese Daten auf drei Arten eingetragen werden: über eine als Geocodierer bezeichnete Funktion, die die Daten aus attributiven Daten ableitet; mit Hilfe anderer Funktionen, die die Daten aus anderen räumlichen Daten ableiten; oder durch Importieren aus Dateien. Dieses Kapitel enthält Beschreibungen zu folgenden Punkten:

- Geocodierung und Verwendung des Control Center zum Geocodieren attributiver Daten im Stapelbetrieb
- Importieren und Exportieren von Daten und Verwendung des Control Center zum Importieren von Daten in Ihr GIS und zum Exportieren aus Ihrem GIS

Informationen zu anderen Funktionen, die neue räumliche Daten aus vorhandenen räumlichen Daten ableiten können, finden Sie im Abschnitt „Funktionen, die neue Geometrien aus vorhandenen generieren" auf Seite 159.

Geocodierer verwenden

In diesem Abschnitt wird der Prozeß der Geocodierung beschrieben und wie ein Geocodierer im Stapelbetrieb vom Control Center aus ausgeführt wird.

Informationen zur Geocodierung

Dieser Abschnitt erläutert die wesentlichen Unterschiede zwischen Geocodierern und ihren Quellen. Außerdem beschreibt er zwei Modi, in denen ein Geocodierer arbeiten kann, und stellt Faktoren vor, die bei der Planung der Arbeit mit einem Geocodierer berücksichtigt werden sollten.

Mit dem DB2 Spatial Extender haben Sie folgende Möglichkeiten:

- Den mit dem DB2 Spatial Extender bereitgestellten Standard-Geocodierer verwenden
- Von anderen Herstellern entwickelte Geocodierer installieren
- Eigene Geocodierer installieren

Der Standard-Geocodierer kann lediglich Adressen in den USA geocodieren und in ST_Point-Daten oder ST_Geometry-Daten umwandeln. Wenn Sie Daten mit anderen räumlichen Datentypen speichern wollen, können Sie einen anderen Geocodierer installieren, der solche Daten generiert. Wenn Sie räumliche Daten benötigen, die Standorte außerhalb der USA darstellen oder Standorte

ohne Adresse, beispielsweise beispielsweise Ackerland mit unterschiedlichem Anteil an Muttererde, können Sie auch für diese Anforderung einen weiteren Geocodierer installieren.

Bevor Sie mit einem zusätzlichen Geocodierer arbeiten können, muß dieser registriert worden sein. Benutzer und Hersteller können einen solchen Geocodierer mit der gespeicherten Prozedur db2gse.gse_register_gc registrieren. Der Geocodierer kann nicht über das Control Center registriert werden. Informationen zu db2gse.gse_register_gc finden Sie im Abschnitt „db2gse.gse_register_gc" auf Seite 94. Allgemeine Informationen zur Verwendung der gespeicherten Prozeduren des DB2 Spatial Extender finden Sie in „Kapitel 9. Gespeicherte Prozeduren" auf Seite 71.

Ein Geocodierer kann in zwei verschiedenen Modi arbeiten:

- Im *Stapelbetrieb* versucht der Geocodierer, in einem einzigen Arbeitsgang alle vorhandenen Quellendaten für räumliche Daten in räumliche Daten umzusetzen und die Spalten mit diesen Daten auszufüllen. Sie können diese Operation vom Fenster **Run Geocoder** aus starten. Alternativ dazu können Sie sie auch in einem Anwendungsprogramm starten, indem Sie das Programm für den Aufruf der gespeicherten Prozedur db2gse.gse_run_gc codieren.
- Im *schrittweise steigenden Modus* wandelt ein Geocodierer Daten um, wenn sie in eine Tabelle eingefügt oder aktualisiert werden. Dabei legt er die resultierenden räumlichen Daten in räumlichen Spalten ab, um die Spalte auf dem neuesten Stand zu halten. Der Geocodierer wird durch Einfüge- und Aktualisierungsauslöser aktiviert; diese Auslöser können Sie über das Fenster **Create Spatial Layer** anfordern. Alternativ dazu können Sie sie in einem Anwendungsprogramm anfordern, indem Sie das Programm für den Aufruf der gespeicherten Prozedur db2gse.gse_enable_autogc codieren.

  Die schrittweise steigende Geocodierung wird auch als *automatische Geocodierung* bezeichnet.

Wenn Sie vorhaben, einen Geocodierer zu verwenden, sollten Sie eventuell die folgenden Faktoren berücksichtigen:

1. Wenn Sie mit dem Control Center arbeiten, verwenden Sie normalerweise das Fenster **Create Spatial Layers**, bevor Sie das Fenster **Run Geocoder** aufrufen. Dies bedeutet, daß Sie den DB2 Spatial Extender anweisen können, mehrere Auslöser für die schrittweise steigende Geocodierung einzurichten, bevor Sie die Geocodierung im Stapelbetrieb starten. Es ist deshalb möglich, daß eine schrittweise Geocodierung einer Geocodierung im Stapelbetrieb vorausgeht. Bei der Verarbeitung aller Quellendaten im Stapelbetrieb werden die gleichen Daten geocodiert wie im schrittweise steigenden Modus. Diese Redundanz führt nicht zu Duplizierungen; wenn räumliche Daten zweimal generiert werden, überschreibt das zweite Datenergebnis das erste.

Die Leistung kann dadurch jedoch sinken. Eine Möglichkeit, diesen Leistungsabfall zu vermeiden, ist das verzögerte Einrichten von Auslösern bis nach der Geocodierung im Stapelbetrieb.

2. Wenn die Auslöser vorhanden sind, wenn Sie mit der Geocodierung im Stapelbetrieb beginnen wollen, empfiehlt es sich, diese Auslöser zu inaktivieren, bis die Geocodierung im Stapelbetrieb abgeschlossen ist. Sie können die Auslöser über das Fenster **Run Geocoder** oder in einem Anwendungsprogramm inaktivieren, indem Sie das Programm für den Aufruf der gespeicherten Prozedur db2gse.gse_disable_autogc codieren. Wenn Sie das Fenster **Run Geocoder** verwenden, reaktiviert der DB2 Spatial Extender die Auslöser automatisch, wenn die Geocodierung abgeschlossen ist. Falls Sie die gespeicherte Prozedur db2gse.gse_disable_autogc verwenden, können Sie die Auslöser über den Aufruf der gespeicherten Prozedur db2gse.gse_enable_autogc reaktivieren.
3. Wenn Sie einen Geocodierer im Stapelbetrieb ausführen wollen, um eine räumliche Spalte mit einem Index auszufüllen, inaktivieren Sie zunächst den Index oder geben Sie ihn frei. Andernfalls bleibt der Index während der Ausführung des Geocodierers betriebsbereit, was die Leistung beträchtlich verschlechtert. Wenn Sie mit dem Control Center arbeiten, können Sie den Index über das Fenster **Run Geocoder** inaktivieren. Der DB2 Spatial Extender reaktiviert den Index automatisch, wenn die Geocodierung abgeschlossen ist. Wenn Sie ein Anwendungsprogramm verwenden, können Sie den Index mit der Anweisung SQL DROP freigeben. Notieren Sie sich in diesem Fall unbedingt die Parameter des Index, so daß Sie ihn nach Abschluß der Geocodierung wieder erstellen können.
4. Wenn der Geocodierer einen Datensatz einliest, versucht er, diesen Datensatz mit seinem Pendant in den Bezugsdaten abzugleichen. Dieser Abgleich muß mindestens eine bestimmte *Genauigkeit* aufweisen, damit der Geocodierer den Datensatz verarbeitet. Eine Genauigkeit von 85 bedeutet beispielsweise, daß ein Quellendatensatz und sein Pendant in den Bezugsdaten zu mindestens 85 Prozent übereinstimmen müssen, damit der Quellendatensatz verarbeitet wird.

   Sie geben selbst an, wie hoch die Genauigkeit sein muß. Denken Sie daran, daß Sie die Genauigkeit eventuell anpassen müssen. Angenommen, die Genauigkeit ist 100. Wenn viele Quellendatensätze Adressen enthalten, die neuer sind als die Bezugsdaten, sind Übereinstimmungen von 100 Prozent zwischen diesen Datensätzen und den Bezugsdaten nicht möglich. Der Geocodierer weist diese Datensätze daher zurück. Insgesamt gilt: Wenn ein Geocodierer räumliche Daten generiert, die ungenügend oder weitgehend ungenau sind, können Sie dieses Problem eventuell umgehen, indem Sie die Genauigkeit ändern und den Geocodierer erneut starten.

Den Geocodierer im Stapelbetrieb ausführen

Dieser Abschnitt enthält eine Übersicht über die Schritte zur Ausführung eines Geocodierers im Stapelbetrieb vom Control Center aus. Im Anschluß an diese Übersicht finden Sie Details zu Durchführung der einzelnen Schritte.

Informationen darüber, welche Berechtigung Sie zum Ausführen eines Geocodierers im Stapelbetrieb benötigen, finden Sie im Abschnitt „Autorisierung" auf Seite 102.

Übersicht über die Schritte zur Ausführung eines Geocodierers im Stapelbetrieb:

1. Öffnen Sie das Fenster **Run Geocoder**.
2. Geben Sie an, welcher Geocodierer verwendet werden soll.
3. Inaktivieren Sie Objekte, die die Leistung des Geocodierers beeinträchtigen könnten.
4. Geben Sie an, wie viele Datensätze geocodiert werden sollen, bevor DB2 die Datensätze festschreibt.
5. Geben Sie an, wie der Geocodierer arbeiten soll.
6. Weisen Sie den DB2 Spatial Extender an, den Geocodierer zu starten.

Ausführliche Schritte zur Ausführung eines Geocodierers im Stapelbetrieb:

1. Öffnen Sie das Fenster **Run Geocoder**.
   a. Heben Sie im Fenster **Control Center** die Komprimierung des Objektbaums auf, bis Sie den Ordner **Tables** in der für die räumlichen Operationen aktivierten Datenbank finden.
   b. Klicken Sie den Ordner **Tables** an. Die Tabellen werden im Inhalts-Teilfenster auf der rechten Seite des Fensters angezeigt.
   c. Klicken Sie mit der rechten Maustaste die Tabelle in dem Inhaltsteilfenster an, und klicken Sie **Spatial layers** im Kontextmenü an. Das Fenster **Spatial Layers** wird geöffnet.
   d. Führen Sie im Fenster **Spatial Layers** folgende Aktionen aus:
      1) Wählen Sie die Schicht aus, die in der auszufüllenden Spalte definiert ist.
      2) Klicken Sie den Druckknopf **Run Geocoder** an. Das Fenster **Run Geocoder** wird geöffnet.
2. Wenn Sie den Standard-Geocodierer verwenden wollen, lassen Sie das Feld **Name**, in dem der Name dieses Standard-Codierers angezeigt wird, unverändert. Wählen Sie andernfalls mit dem Feld den gewünschten Geocodierer aus.

3. Inaktivieren Sie Objekte, die die Leistung des Geocodierers beeinträchtigen könnten:
   - Wenn die auszufüllende Spalte einen Index hat, wählen Sie das Markierungsfeld **Temporarily disable spatial indexes during geocoding process** aus.
   - Wenn Auslöser zum Aktivieren der schrittweise steigenden Geocodierung für diese Spalte festgelegt wurden, wählen Sie das Markierungsfeld **Temporarily disable spatial triggers during geocoding process** aus.

     Der Index und die Auslöser werden automatisch reaktiviert, wenn Sie **OK** im Fenster **Run Geocoder** anklicken.
4. Geben Sie mit dem Drehknopf **Commit scope** an, wie viele Datensätze geocodiert werden sollen, bevor DB2 die Datensätze festschreibt. Wenn DB2 beispielsweise 100 geocodierte Datensätze in einem Arbeitsgang festschreiben soll, geben Sie die Zahl 100 ein.

   Tip: Wenn DB2 die Datensätze erst festschreiben soll, nachdem alle Datensätze verarbeitet wurden, geben Sie Null ein.
5. Geben Sie mit den Feldern in der Auswahlgruppe **Geocoder parameters** an, wie der Geocodierer arbeiten soll:
   - Geben Sie mit dem Drehknopf **Precision level** an, wie genau (in Prozent) die Übereinstimmung zwischen den Quellendatensätzen und den entsprechenden Bezugsdaten sein soll. Weitere Informationen zur Genauigkeit finden Sie im Abschnitt „Informationen zur Geocodierung" auf Seite 43.
   - Wenn Sie einen Geocodierer eines anderen Anbieters verwenden und mit den von diesem Geocodierer unterstützten Merkmalen arbeiten wollen, legen Sie diese Merkmale über das Feld **Properties** fest.
   - Wenn nur ein Teil der Zeilen in der ausgewählten Tabelle geocodiert werden soll, wählen Sie mit dem Feld **WHERE clause** eine Klausel SELECT WHERE aus, die die Kriterien für die gewünschten Zeilen angibt. Diese Klausel kann auf beliebige Spalten in der Tabelle verweisen.

     Geben Sie nur die Kriterien ein. Lassen Sie das Schlüsselwort WHERE weg. Beispiel: Wenn die Tabelle eine Spalte STATE enthält und nur die Zeilen geocodiert werden sollen, die in dieser Spalten den Wert MA enthalten, geben Sie ein:

     ```
     STATE='MA'
     ```
6. Klicken Sie **OK** an, um den Geocodierer auszuführen.

Daten importieren und exportieren

Dieser Abschnitt beschreibt die Prozesse zum Importieren und Exportieren von Daten sowie die Verwendung des Control Center für folgende Aufgaben:

- Importieren von Daten aus einer Datenaustauschdatei in eine neue oder vorhandene Tabelle
- Importieren von Daten aus einer Datenaustauschdatei in eine vorhandene Tabelle
- Exportieren von Daten aus einer Tabelle in eine Datenaustauschdatei

Informationen zum Importieren und Exportieren

Dieser Abschnitt listet Gründe für das Importieren oder Exportieren räumlicher Daten auf. Außerdem enthält er eine Beschreibung der Datenaustauschdateienen, die als Schnittstelle zwischen der Quelle und dem Ziel des Importvorgangs dienen.

Sie können mit dem DB2 Spatial Extender räumliche Daten aus Datenaustauschdateien importieren oder in Datenaustauschdateien exportieren. Betrachten Sie die folgenden Szenarien:

- Ihr GIS enthält räumliche Daten, die Ihre Büros, Ihre Kunden und andere Geschäftsdaten darstellen. Sie wollen diese Daten mit räumlichen Daten zur Infrastrukturumgebung Ihrer Organisation ergänzen — Städte, Straßen, Verkehrsknotenpunkte etc. Die gewünschten Daten sind über einen Anbieter von Karten verfügbar. Sie können mit dem DB2 Spatial Extender diese Daten aus einer von dem Anbieter bereitgestellten Datenaustauschdatei importieren.
- Sie wollen räumliche Daten aus einem Oracle-System in Ihr DB2 Spatial Extender-GIS migrieren. Zunächst laden Sie mit einem Oracle Dienstprogramm die Daten in eine Datenaustauschdatei. Anschließend importieren Sie mit dem DB2 Spatial Extender die Daten aus dieser Datei in die Datenbank, die Sie für räumliche Operationen aktiviert haben.
- Sie wollen mit einem GIS-Browser visuelle Darstellungen von räumlichen Informationen zu Kunden anzeigen. Der Browser benötigt lediglich Daten, mit denen er arbeiten kann; eine Verbindung zu einer Datenbank ist nicht erforderlich. Sie können mit dem DB2 Spatial Extender die Daten in eine Datenaustauschdatei exportieren und anschließend mit einem Browser-Dienstprogramm die Daten in den Browser laden.

Das Control Center unterstützt zwei Arten von Datenaustauschdateien für den DB2 Spatial Extender: Formdateien und ESRI_SDE-Transferdateien. Formdateien werden häufig zum Importieren von Daten aus Dateisystemen verwendet und zum Exportieren von Daten in Dateien, die in Dateisysteme geladen werden sollen. ESRI_SDE-Transferdateien werden häufig zum Importieren von Daten aus ESRI-Datenbanken verwendet.

Daten in eine neue oder eine vorhandene Tabelle importieren

Dieser Abschnitt enthält eine Übersicht über die Schritte zum Importieren von Daten aus einer Form- oder ESRI_SDE-Transferdatei in eine neue oder bereits vorhandene Tabelle. Im Anschluß an diese Übersicht finden Sie Details zu Durchführung der einzelnen Schritte.

Informationen darüber, welche Berechtigungen zum Importieren von Formdaten erforderlich sind, finden Sie im Abschnitt „Autorisierung" auf Seite 92. Informationen darüber, welche Berechtigungen zum Importieren von ESRI_SDE-Daten erforderlich sind, finden Sie im Abschnitt „Autorisierung" auf Seite 90.

Übersicht über die Schritte zum Importieren von Daten in eine neue oder vorhandene Tabelle:

1. Öffnen Sie das Fenster **Import Spatial Data**.
2. Geben Sie den Pfad, den Namen und das Format der Datei an, die die zu importierenden Daten enthält.
3. Geben Sie an, wie viele Datensätze vor jedem Festschreiben geocodiert werden sollen.
4. Wenn Sie räumliche Daten in eine zu erstellende Tabelle importieren wollen, geben Sie einen Namen für diese Tabelle und einen Namen für die Spalte, in der die Daten gespeichert werden sollen, an. Wenn Sie räumliche Daten in eine vorhandene Tabelle importieren, geben Sie an, in welcher Spalte die Daten gespeichert werden sollen.
5. Geben Sie an, welches räumliche Bezugssystem den Daten zugeordnet werden soll.
6. Geben Sie eine Datei an, in der die Datensätze gesammelt werden sollen, die nicht importiert werden konnten.
7. Weisen Sie den DB2 Spatial Extender an, die Daten zu importieren und, falls Sie über dieses Fenster eine Tabelle definiert hatten, eine Tabelle zu erstellen und die Spalte, für die die Daten bestimmt sind, als Schicht zu registrieren.

Details zu den Schritten zum Importieren von Daten in eine neue oder vorhandene Tabelle:

1. Öffnen Sie das Fenster **Import Spatial Data**.
   a. Heben Sie im Fenster **Control Center** die Komprimierung des Objektbaums auf, bis Sie den Ordner **Databases** unter dem Server finden, auf dem der DB2 Spatial Extender ausgeführt wird.
   b. Klicken Sie den Ordner **Databases** an. Die Datenbanken werden im Inhalts-Teilfenster auf der rechten Seite des Fensters angezeigt.

c. Klicken Sie mit der rechten Maustaste die Datenbank an, in die die Daten importiert werden sollen, und klicken Sie **Spatial Extender —> Import Spatial Data** im Kontextmenü an. Das Fenster **Import Spatial Data** wird geöffnet.

2. Geben Sie den Pfad, den Namen und das Format der Datei an, die die zu importierenden Daten enthält:
   a. Geben Sie im Feld **File name** den Pfad und den Namen der Datei an.
   b. Geben Sie das Format im Feld **File format** an. Das Format kann eines der folgenden sein:

      Shape Dies ist der Standardwert.

      ESRI_SDE
      Wenn Sie dieses Format angeben, enthält das Feld **Spatial reference name** standardmäßig den Namen des räumlichen Bezugssystems, das diesem Format zugeordnet ist.

3. Geben Sie im Feld **Commit scope** die Anzahl der Datensätze an, die vor jedem Festschreiben importiert werden sollen. Wenn DB2 beispielsweise 100 Datensätze in einem Arbeitsgang festschreiben soll, geben Sie die Zahl 100 ein.

   Tip: Wenn DB2 die Datensätze erst festschreiben soll, nachdem alle Datensätze verarbeitet wurden, geben Sie Null ein.

4. Geben Sie die Tabelle und die Spalte an, für die die Daten bestimmt sind.
   a. Geben Sie im Feld **Layer schema** das Schema für die Tabelle an, in die die Daten importiert werden sollen.
   b. Geben Sie die Tabelle und die Spalte an:
      - Wenn die Tabelle noch nicht vorhanden ist:
        1) Geben Sie im Feld **Layer table** einen Namen für die Tabelle ein.
        2) Geben Sie im Feld **Layer column** einen Namen für die Spalte ein, die die importierten Daten enthalten soll. Der DB2 Spatial Extender registriert diese Spalte automatisch als Schicht.
      - Wenn die Tabelle bereits vorhanden ist:
        1) Geben Sie im Feld **Layer table** die Tabelle an. Die Tabelle muß bereits die Spalte enthalten, in der die importierten Daten gespeichert werden sollen. Darüber hinaus muß diese Tabellenspalte bereits als Schicht registriert worden sein.
        2) Geben Sie im Feld **Layer column** den Namen der Spalte an, für die die importierten Daten bestimmt sind.

5. Geben Sie im Feld **Spatial reference name** das räumliche Bezugssystem ein, das diesen Daten zugeordnet werden soll bzw. wählen Sie dieses Bezugssystem aus. (Wenn die Daten aus einer ESRI_SDE-Transferdatei stammen, wird der Name des zugeordneten räumlichen Bezugssystems automatisch in dem Feld angezeigt.)
6. Geben Sie im Feld **Exception file** den Pfad und den Namen für eine neue Datei ein, in der die Datensätze gesammelt werden können, die nicht importiert werden konnten. Später können Sie diese Datensätze korrigieren und aus dieser Datei importieren.

   DB2 Spatial ExtenderDer erstellt diese Datei; Sie brauchen keine bereits vorhandene Datei auszuwählen.
7. Klicken Sie **OK** an, um die Daten zu importieren. Wenn Sie einen Namen für eine Tabelle angegeben haben, die noch nicht vorhanden ist, wird diese Tabelle ebenfalls erstellt, und die Spalte, für die die Daten bestimmt sind, wird als Schicht registriert. Darüber hinaus wird die angegebene Ausnahmedatei erstellt.

Daten in eine vorhandene Tabelle importieren

Dieser Abschnitt enthält eine Übersicht über die Schritte zum Importieren von Daten aus einer Form- oder ESRI_SDE-Transferdatei in eine bereits vorhandene Tabelle. Im Anschluß an diese Übersicht finden Sie Details zu Durchführung der einzelnen Schritte.

Informationen darüber, welche Berechtigungen zum Importieren von Formdaten erforderlich sind, finden Sie im Abschnitt „Autorisierung" auf Seite 92. Informationen darüber, welche Berechtigungen zum Importieren von ESRI_SDE-Daten erforderlich sind, finden Sie im Abschnitt „Autorisierung" auf Seite 90.

Übersicht über die Schritte zum Importieren von Daten in eine vorhandene Tabelle:

1. Öffnen Sie das Fenster **Import Spatial Data**.
2. Geben Sie den Pfad und den Namen der Datei an, die die zu importierenden Daten enthält.
3. Geben Sie an, wie viele Datensätze vor jedem Festschreiben geocodiert werden sollen.
4. Geben Sie die Spalte an, die die zu importierenden räumlichen Daten enthält.
5. Geben Sie an, welches räumliche Bezugssystem diesen Daten zugeordnet werden soll.

6. Geben Sie eine Datei an, in der Datensätze gesammelt werden sollen, die nicht importiert werden konnten.
7. Weisen Sie den DB2 Spatial Extender an, die Daten zu importieren und, falls Sie eine bisher noch nicht vorhandene Spalte definiert hatten, diese Spalte zu erstellen und als Schicht zu registrieren.

Details zu den Schritten zum Importieren von Daten in eine vorhandene Tabelle:

1. Öffnen Sie das Fenster **Import Spatial Data**.
   a. Heben Sie im Fenster **Control Center** die Komprimierung des Objektbaums auf, bis Sie den Ordner **Tables** für die Datenbank finden, in die Sie Daten importieren wollen.
   b. Klicken Sie den Ordner **Tables** an. Die Tabellen werden im Inhalts-Teilfenster auf der rechten Seite des Fensters angezeigt.
   c. Klicken Sie mit der rechten Maustaste die Tabelle an, in die die Daten importiert werden sollen, und klicken Sie **Spatial Extender —> Import Spatial Data** im Kontextmenü an. Das Fenster **Import Spatial Data** wird geöffnet.
2. Geben Sie im Feld **File name** den Pfad und den Namen der Datei an, die die zu importierenden Daten enthält.
3. Geben Sie im Feld **Commit scope** die Anzahl der Datensätze an, die vor jedem Festschreiben importiert werden sollen. Wenn DB2 beispielsweise 100 Datensätze in einem Arbeitsgang festschreiben soll, geben Sie die Zahl 100 ein.

   Tip: Wenn DB2 die Datensätze erst festschreiben soll, nachdem alle Datensätze verarbeitet wurden, geben Sie Null ein.
4. Geben Sie die Spalte an, die die zu importierenden räumlichen Daten enthält.
   - Wenn die Spalte in der Tabelle noch nicht vorhanden ist, geben Sie im Feld **Layer column** einen Namen für die Spalte ein.
   - Wenn die Spalte bereits vorhanden ist, wählen Sie den Namen der Spalte im Feld **Layer column** aus bzw. geben Sie ihn ein.
5. Geben Sie im Feld **Spatial reference name** das räumliche Bezugssystem ein, das den importierten Daten zugeordnet werden soll.
   - Wenn Sie einer Tabelle eine Spalte hinzufügen, geben Sie den Namen des räumlichen Bezugssystems ein bzw. wählen Sie ihn aus.
   - Wenn die importierten Daten für eine bereits vorhandene Spalte bestimmt sind, lassen Sie das Feld **Spatial reference name** unverändert. Das Feld zeigt den Namen des standardmäßigen räumlichen Bezugssystems an.

6. Geben Sie im Feld **Exception file** den Pfad und den Namen für eine neue Datei ein, in der die Datensätze gesammelt werden können, die nicht importiert werden konnten. Später können Sie diese Datensätze korrigieren und aus dieser Datei importieren.

   Der DB2 Spatial Extender erstellt diese Datei; Sie brauchen keine bereits vorhandene Datei auszuwählen.
7. Klicken Sie **OK** an, um die Daten zu importieren. Wenn Sie eine noch nicht vorhandene Spalte angegeben haben, wird diese Spalte erstellt und als Schicht registriert. Darüber hinaus wird die angegebene Ausnahmedatei erstellt.

Daten in eine Formdatei exportieren

Dieser Abschnitt enthält eine Übersicht über die Schritte zum Exportieren von Daten in eine Formdatei. Im Anschluß an diese Übersicht finden Sie Details zu Durchführung der einzelnen Schritte.

Informationen darüber, welche Berechtigungen für diese Schritte erforderlich sind, finden Sie im Abschnitt „Autorisierung" auf Seite 88.

Übersicht über die Schritte zum Exportieren von Daten in eine Formdatei:

1. Öffnen Sie das Fenster **Export Spatial Data**.
2. Geben Sie die Spalte an, die die zu exportierenden räumlichen Spalten enthält.
3. Wenn Sie eine Untermenge der Datenzeilen exportieren wollen, kennzeichnen Sie diese Untermenge für den DB2 Spatial Extender.
4. Geben Sie den Pfad und den Namen der Datei an, in die Sie Daten exportieren wollen.
5. Weisen Sie den DB2 Spatial Extender an, die Daten zu exportieren.

Details zu den Schritten zum Exportieren von Daten in eine Formdatei:

1. Öffnen Sie das Fenster **Export Spatial Data**.
   a. Heben Sie im Fenster **Control Center** die Komprimierung des Objektbaums auf, bis Sie den Ordner **Tables** oder den Ordner **Views** in der Datenbank finden, die die räumlichen Daten enthält:
   b. Klicken Sie den Ordner **Tables** oder **Views** an. Tabellen oder Sichten werden im Inhalts-Teilfenster auf der rechten Seite des Fensters angezeigt.
   c. Klicken Sie mit der rechten Maustaste die Tabelle oder Sicht an, die die zu exportierenden Daten enthält, und klicken Sie **Spatial Extender —> Export Spatial Data** im Kontextmenü an. Das Fenster **Export Spatial Data** wird geöffnet.
2. Geben Sie im Feld **Layer column** den Namen der Spalte an, die die zu exportierenden räumlichen Daten enthält.

3. Wenn Sie nur eine Untermenge der Tabellenzeilen exportieren wollen, geben Sie im Feld **WHERE clause** eine WHERE-Klausel ein, die die Kriterien für die gewünschten Zeilen angibt. Sie können in dieser Klausel nur auf Spalten in der Tabelle oder Sicht verweisen, aus der Sie Daten exportieren.

   Geben Sie nur die Kriterien ein. Lassen Sie das Schlüsselwort WHERE weg. Beispiel: Wenn die Tabelle bzw. Sicht eine Spalte STATE enthält und nur die Zeilen geocodiert werden sollen, die in dieser Spalten den Wert MA enthalten, geben Sie ein:

   ```
   STATE='MA'
   ```

4. Geben Sie im Feld **File name** den Pfad und den Namen der Datei ein, in die Sie Daten exportieren.
5. Klicken Sie **OK** an, um die Daten zu exportieren.

Kapitel 6. Räumliche Indizes erstellen

In diesem Kapitel wird beschrieben, wie über das Control Center ein Index für Ihre räumlichen Daten erstellt wird.

Nachdem Sie die räumlichen Daten mit Ihren Daten ausgefüllt haben, können Sie einen räumlichen Index erstellen. Typische Indexstrukturen wie beispielsweise die B-Baumstruktur führen lineare, eindimensionale Sortierungen für Tabellendaten durch. Für räumliche Operationen aktivierte Tabellendaten werden nicht als einzelner Eintrag gespeichert, sondern sind zweidimensional. Räumliche Geometrien wie Polygone bestehen beispielsweise aus verschiedenen Koordinatenwerten in einer räumlichen Spalte oder Schicht. Da ein B-Baumstruktur-Index keine Typen von räumlichen Daten verarbeiten kann, hat der DB2 Spatial Extender eine spezielle Indexierungstechnologie erstellt, die als *Gitterindex* bezeichnet wird. Dieser Gitterindex basiert auf dem B-Baumstruktur-Index, der entsprechend erweitert wurde, um auch zweidimensionale Daten verarbeiten und Indexierungen mit räumlichen Spalten ausführen zu können. Der Gitterindex unterstützt drei Schichten und ermöglicht eine hohe Leistung mit einer Vielzahl verschiedener Objekte, Größen und Verteilungen von Daten. Weitere Informationen zu räumlichen Indizes finden Sie in „Kapitel 12. Räumliche Indizes" auf Seite 119.

Informationen darüber, welche Berechtigungen zum Erstellen eines räumlichen Index erforderlich sind, finden Sie im Abschnitt „Autorisierung" auf Seite 83.

Über das Control Center einen räumlichen Index erstellen

So erstellen Sie über das Control Center einen räumlichen Index:

1. Wählen Sie im Objektbaum den Ordner **Tables** aus. Alle vorhandenen Tabellen werden im Inhalts-Teilfenster angezeigt.
2. Klicken Sie im Inhalts-Teilfenster mit der rechten Maustaste die Tabelle an, für die Sie einen Index erstellen wollen, und klicken Sie **Spatial Extender —> Spatial Indexes** im Kontextmenü an. Das Fenster **Spatial Indexes** wird geöffnet.
3. Klicken Sie im Fenster **Spatial Indexes** die Option **Create** an. Das Fenster **Create Spatial Index** wird geöffnet.
4. Geben Sie im Feld **Name** den Namen des neuen räumlichen Index ein, den Sie erstellen wollen.

Anmerkung: Sie brauchen kein Schema anzugeben. Der DB2 Spatial Extender fügt das Schema automatisch ein und erstellt einen vollständig qualifizierten Namen dafür.

5. Wählen Sie im Feld **Layer column** die Schicht aus, für die Sie einen Index erstellen wollen.

   Eine Schicht ist eine räumliche Spalte, die für den DB2 Spatial Extender definiert oder registriert wurde.

6. Geben Sie in den Feldern **Grid size** die Größe des Gitters ein, die Sie den einzelnen Feldern zuordnen wollen.

   Die Gitterstufen **Finest**, **Middle** und **Coarsest** werden durch Vergrößern der Zellengröße eingegeben. Somit muß die zweite Schicht eine höhere Zellengröße haben als die erste und die dritte Schicht eine höhere als die zweite.

Größe der Gitterzellen festlegen

Die richtige Gittergröße wird über ″Versuch und Irrtum″ festgelegt. Es wird empfohlen, die Gittergröße in Relation zur ungefähren Größe des Objekts, das indexiert werden soll, festzulegen. Gittergrößen, die zu groß oder zu klein festgelegt wurden, können zu einer Verschlechterung der Leistung führen. Wenn die Größe zu klein festgelegt wird, wirkt sich dies auf das Verhältnis Schlüssel/Objekt bei der Indexsuche aus. In diesem Fall werden zu viele Schlüssel erstellt, und es wird eine große Anzahl Kandidaten zurückgegeben. Für Gittergrößen, die zu groß gesetzt wurden, gibt der Anfangsindex eine kleine Anzahl von Kandidaten zurück, aber die Leistung kann sich bei der endgültigen Tabellensuche verschlechtern.

Weitere Informationen zur Auswahl der Größe von Gitterzellen und der Anzahl von Gitterstufen finden Sie im Abschnitt „Gitterzellengröße auswählen” auf Seite 126.

Kapitel 7. Räumliche Informationen abrufen und analysieren

Nach dem Konstruieren der räumlichen Indizes stehen die räumlichen Tabellen zum Arbeiten zur Verfügung. In diesem Kapitel werden Informationen zum Abrufen und Analysieren räumlicher Daten beschrieben. Das Kapitel enthält eine Übersicht über die verschiedenen Methoden zum Abrufen sowie Beispiele zu Tabellenabfragen, die räumliche Funktionen verwenden.

Methoden zum Ausführen einer räumlichen Analyse

Sie können eine räumliche Analyse mit SQL und räumlichen Funktionen mit einer der folgenden Programmierumgebungen durchführen:

- Mit einem Geobrowser (z. B. ESRI ArcExplorer).

  Weitere Information zur Verwendung des ArcExplorer finden Sie unter *Using ArcExplorer*; dieses Dokument ist über die ESRI Web-Site verfügbar unter der Adresse http://www.esri.com.
- Mit interaktiven SQL-Anweisungen
- Mit benutzerentwickelten Anwendungen (z. B. ODBC, JDBC und eingebettetem SQL).

Anwendungen können über das DB2 Command Center, das DB2-Befehlsfenster oder den Befehlszeilenprozessor gestartet werden.

Räumliche Abfrage erstellen

In diesem Abschnitt wird das Erstellen räumlicher Abfragen, die räumliche Funktionen und Prädikate verwenden, beschrieben.

Räumliche Funktionen und SQL

Der DB2 Spatial Extender umfaßt Funktionen, die verschiedene Operationen mit räumlichen Daten ausführen. Die Beispiele in diesem Abschnitt zeigen, wie Sie mit den räumlichen Funktionen eigene räumliche Abfragen erstellen können.

Tabelle 3 zeigt eine Liste der räumlichen Funktionen und der Arten von Operationen, die Sie ausführen können.

Tabelle 3. Räumliche Funktionen und Operationen

Funktionstyp	Beispiel zur Operation
Berechnung	Berechnet die Entfernung zwischen zwei Punkten

Tabelle 3. Räumliche Funktionen und Operationen (Forts.)

Funktionstyp	Beispiel zur Operation
Vergleich	Findet alle Kunden innerhalb eines Überflutungsbereichs
Datenaustausch	Umwandeln von Daten in unterstützte Formate
Umsetzung	Hinzufügen eines Fünf-Meilen-Radius zu einem Punkt

Weitere Informationen zu räumlichen Funktionen finden Sie in den Abschnitten „Kapitel 13. Geometrien und zugeordnete räumliche Funktionen" auf Seite 129 und „Kapitel 14. Räumliche Funktionen für SQL-Abfragen" auf Seite 169.

Beispiel 1: Vergleich

Die folgende Abfrage findet die durchschnittliche Entfernung vom Kunden bis zum Kaufhaus. Die in diesem Beispiel verwendeten räumlichen Funktionen sind ST_Distance and 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
```

Beispiel 2: Datenaustausch

Die folgende Abfrage findet die Kundenstandorte für die Kunden in der Gegend der San Francisco Bay. Die in diesem Beispiel verwendeten räumlichen Funktionen sind ST_AsText (Datenaustausch) und ST_Within. ST_AsText wandelt die räumlichen Daten in der Spalte c.location in das Format OGC TEXT um.

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

Beispiel 3: Berechnung

Die folgende Abfrage findet alle Straßen, die länger als 10,5 Meilen sind. Die in diesem Beispiel verwendete räumliche Funktion ist ST_Length.

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

Beispiel 4: Umsetzung

Diese Abfrage findet die Kunden, die in einem Überflutungsgebiet oder innerhalb von 2 Meilen von den Grenzen eines Überflutungsgebiets wohnen. Die in diesem Beispiel verwendeten räumlichen Funktionen sind ST_Buffer (Umsetzung) und ST_Within.

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

Räumliche Prädikate und SQL

Mit einer speziellen Gruppe räumlicher Funktionen, die als räumliche Prädikate bezeichnet werden, kann die Leistung gesteigert werden. Räumliche Prädikate wie beispielsweise ST_Overlaps, das zwei Polygone vergleicht, um festzustellen, ob sie sich überlappen, können im Hinblick auf Zeit- und Speicheranforderungen sehr aufwendig sein. Optimierungstechniken zum Minimieren der Ausführungskosten sind daher von großer Bedeutung. Der DB2 Query Optimizer verwendet den räumlichen Index zum Verbessern der Abfrageleistung, wenn Sie mit den räumlichen Prädikaten entsprechend den weiter hinten in diesem Kapitel beschriebenen Regeln arbeiten. Weitere Informationen zu räumlichen Prädikaten finden Sie im Abschnitt „Prädikatfunktionen" auf Seite 145. Die zum Nutzen des räumlichen Index verwendeten räumlichen Prädikate sind:

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

Eine vollständige Liste alle räumlichen Funktionen und Prädikate finden Sie in „Kapitel 14. Räumliche Funktionen für SQL-Abfragen" auf Seite 169.

Regeln zur Nutzung des Index

Die folgenden Regeln gelten, wenn Sie räumliche Abfragen mit räumlichen Prädikaten optimieren wollen:

- Das Prädikat muß in der WHERE-Klausel verwendet werden.
- Das Prädikat muß auf der linken Seite des Vergleichs stehen. Beispiel:

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

- Bei Vergleichen auf Gleichheit ist die ganzzahlige Konstante 1 zu verwenden.

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

- In dem Prädikat muß eine räumliche Spalte als Suchziel verwendet werden, und es muß mit dieser Spalte ein räumlicher Index erstellt werden.

Beispiele zur Nutzung des Index

Tabelle 4 zeigt richtige und falsche Beispiele zum Erstellen räumlicher Abfragen nur Nutzung des räumlichen Index.

Tabelle 4. Regeln zur Nutzung des Index

Räumliche Abfrage	Regel verletzt
`SELECT * FROM customers c WHERE db2gse.ST_Within(c.location, :BayArea)=1`	In diesem Beispiel wurde keine Regel verletzt.
`SELECT * FROM customers c WHERE db2gse.ST_Distance(c.location, :SanJose)<10`	In diesem Beispiel wurde keine Regel verletzt.
`SELECT * FROM customers c WHERE db2gse.ST_Length(c.location)>10`	Das Prädikat muß in der WHERE-Klausel verwendet werden. (ST_Length ist eine räumliche Funktion, aber *kein* Prädikat.)
`SELECT * FROM customers c WHERE 1=db2gse.ST_Within(c.location, :BayArea)`	Das Prädikat muß auf der linken Seite des Vergleichs stehen.
`SELECT * FROM customers c WHERE db2gse.ST_Within(c.location, :BayArea)=2`	Bei Vergleichen auf Gleichheit ist die ganzzahlige Konstante 1 zu verwenden.
`SELECT * FROM customers c WHERE db2gse.ST_Within(:SanJose, :BayArea)=1`	In dem Prädikat muß eine räumliche Spalte als Suchziel verwendet werden, und es muß mit dieser Spalte ein räumlicher Index erstellt werden. (SanJose und BayArea sind keine räumlichen Spalten, daher kann ihnen kein räumlicher Index zugeordnet werden.)

Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben

In diesem Kapitel wird beschrieben, wie mit dem DB2 Spatial Extender-Beispielprogramm Anwendungen zum Arbeiten und Anpassen räumlicher Informationen geschrieben werden. Folgende Punkte werden beschrieben:

- Beispielprogramm verwenden
- Die Schritte des Beispielprogramms

Das Beispielprogramm verwenden

Das DB2 Spatial Extender-Beispielprogramm vereinfacht das Schreiben von Anwendungen. Mit dem Beispielprogramm haben Sie folgende Möglichkeiten:

- Räumliche Routineaufgaben automatisieren
- Mustercode ausschneiden und in eigene Anwendungen einfügen
- Erforderliche Schritte zum Erstellen und Verwalten einer für räumliche Funktionen aktivierten Datenbank verstehen

Verwenden Sie das Beispielprogramm zum Codieren komplexer Aufgaben für den DB2 Spatial Extender, beispielsweise zum Schreiben einer Anwendung, die eine Datenbankschnittstelle zum Aufruf gespeicherter DB2 Spatial Extender-Funktionen verwendet. Mit dem Beispielprogramm können Sie Ihre Anwendungen kopieren und anpassen. Wenn Sie mit den Schritten zur Programmierung des DB2 Spatial Extender nicht vertraut sind, können Sie das Beispielprogramm ausführen, in dem die einzelnen Schritte ausführlich beschrieben sind. Zunächst müssen Sie jedoch das Beispielprogramm erstellen. Hierzu verwenden Sie die Beispiel-Make-Datei. Anweisungen zum Erstellen und Ausführen des Beispielprogramms finden Sie im Abschnitt „Installation überprüfen" auf Seite 22.

Die Schritte des Beispielprogramms

Tabelle 5 auf Seite 62 zeigt die Schritte des Beispielprogramms, die zugeordneten gespeicherten Prozeduren und eine Beschreibung der einzelnen Schritte. Die C-Funktionen zum Aufrufen der gespeicherten Prozeduren sind in der Spalte "Aktion" von Tabelle 5 auf Seite 62 angegeben und in Klammern eingeschlossen. Weitere Informationen zu gespeicherten Prozeduren finden Sie in „Kapitel 9. Gespeicherte Prozeduren" auf Seite 71. Das Beispielprogramm basiert auf den im Abschnitt „Szenario: Eine Versicherungsgesellschaft aktualisiert ihr GIS" auf Seite 14 vorgestellten Szenarien.

Tabelle 5. DB2 Spatial Extender-Beispielprogramm

Schritte des Beispiel-programms	Aktion	Beschreibung
Aktivieren/Inaktivieren der räumlichen Datenbank	1. Aktivieren der räumlichen Datenbank (gseEnableDB) 2. Inaktivieren der räumlichen Datenbank (gseDisableDB) 3. Aktivieren der räumlichen Datenbank (gseEnableDB)	1. Dies ist der erste erforderliche Schritt zur Verwendung des DB2 Spatial Extender. Eine für räumliche Operationen aktivierte Datenbank enthält spezielle Typen, eine Gruppe räumlicher Funktionen, einen neuen Indextyp und eine Gruppe von Verwaltungstabellen und Sichten. 2. Dieser Schritt wird normalerweise ausgeführt, wenn Sie die räumlichen Funktionen für die falsche Datenbank aktiviert haben. Wenn Sie eine räumliche Datenbank aktivieren, entfernen Sie eine Gruppe räumlicher Typen, eine Gruppe räumlicher Prädikate, einen neuen Indextyp und eine Gruppe von Verwaltungstabellen und Sichten. **Anmerkung:** Das Inaktivieren der Datenbank schlägt fehl, wenn Objekte erstellt wurden, die von den Objekten abhängen, die beim Aktivieren der Datenbank erstellt wurden. Wenn Sie beispielsweise eine Tabelle mit einer räumlichen Spalte des Typs ST_Point erstellen, schlägt das Inaktivieren der Datenbank fehl. Dieser Fehler tritt auf, da die Tabelle vom Typ ST_Point abhängt, der durch das Inaktivieren der Datenbank freigegeben werden soll. 3. Wie 1.

Tabelle 5. DB2 Spatial Extender-Beispielprogramm (Forts.)

Schritte des Beispiel-programms	Aktion	Beschreibung
Räumliche Bezugs-systeme registrieren	1. Registrieren des räumlichen Bezugssystems für die Spalte LOCATION der Tabelle CUSTOMERS (gseEnableSref) 2. Registrieren des räumlichen Bezugssystems für die Spalte LOCATION der Tabelle OFFICES (gseEnableSref) 3. Registrierung des räumlichen Bezugssystems für die Spalte LOCATION der Tabelle OFFICES aufheben (gseDisableSref) 4. Erneut Registrieren des räumlichen Bezugssystems für die Spalte ZONE der Tabelle OFFICES (gseEnableSref)	1. Dieser Schritt definiert ein neues räumliches Bezugssystem (SRS), das zum Interpretieren der räumlichen Daten der Tabelle CUSTOMERS verwendet werden soll. Ein räumliches Bezugssystem enthält Geometriedaten in einer Form, die in einer Spalte einer räumlich aktivierten Daten-bank gespeichert werden kann. Nachdem das SRS für eine bestimmte Schicht registriert wurde, können die entsprechenden Koordinaten für diese Schicht in der zugeordneten Spalte der Tabelle CUSTOMERS gespeichert werden. 2. Dieser Schritt definiert ein neues räumliches Bezugssystem (SRS), das zum Interpretieren der räumlichen Daten der Schicht von OFFICES verwendet werden soll. Für jede Tabellenschicht muß ein SRS definiert worden sein. Die Schichten der Tabelle OFFICES erfordern eventuell die Zuordnung eines anderen SRS als für die Schichten der Tabelle CUSTOMERS. 3. Dieser Schritt wird ausgeführt, wenn Sie die falschen SRS-Parameter für die Schicht oder die räumliche Spalte angeben. Wenn Sie die Registrierung eines SRS für die Tabelle OFFICES aufheben, entfernen Sie die Definition mit ihren zugeordneten Parametern. 4. Dieser Schritt definiert ein neues räumliches Bezugssystem (SRS), das zum Interpretieren der räumlichen Daten der Schicht von OFFICES verwendet werden soll.

Tabelle 5. DB2 Spatial Extender-Beispielprogramm (Forts.)

Schritte des Beispiel-programms	Aktion	Beschreibung
Räumliche Tabellen erstellen	1. Ändern der Tabelle CUSTOMERS durch Hinzufügen der Spalte LOCATION (gseSetupTables) 2. Erstellen der Tabelle OFFICES (gseSetupTables)	1. Die Tabelle CUSTOMERS stellt Geschäfts-daten dar, die seit mehreren Jahren in der Datenbank gespeichert wurden. Die Anweisung ALTER TABLE fügt eine neue Spalte (LOCATION) des Typs ST_Point hinzu. Diese Spalte wird durch eine Geocodierung der Adreßspalten in einem späteren Schritt ausgefüllt. 2. Die Tabelle OFFICES stellt neben anderen Daten die Vertriebszone für jede Niederlassung eines Versicherungs-unternehmens dar. Die gesamte Tabelle wird in einem späteren Schritt mit den attributiven Daten aus einer Nicht-DB2-Datenbank ausgefüllt. Dieser Schritt umfaßt das Importieren attributiver Daten in die Tabelle OFFICES auf einer Datei SHAPE.
Registrieren der räumlichen Schichten	1. Registrieren der Spalte LOCATION in der Tabelle CUSTOMERS als Schicht (gseRegisterLayer) 2. Registrieren der Spalte ZONE der Tabelle OFFICES als Schicht (gseRegisterLayer)	Mit diesen Schritten werden die Spalten LOCATION und ZONE als Schichten im DB2 Spatial Extender registriert. Bevor eine räumliche Spalte ausgefüllt oder über die DB2 Spatial Extender-Dienstprogramme (z. B. den Geocodierer) aufgerufen werden kann, müssen Sie sie als Schicht registrieren.

Tabelle 5. DB2 Spatial Extender-Beispielprogramm (Forts.)

Schritte des Beispiel-programms	Aktion	Beschreibung
Ausfüllen der räumlichen Schichten	1. Geocodieren der Adreßdaten für die Spalte LOCATION der Tabelle CUSTOMERS (gseRunGC) 2. Laden der Tabelle OFFICES mit dem Append-Modus (gseImportShape) 3. Laden der Tabelle HAZARD_ZONE mit dem Create-Modus (gseImportShape)	1. Mit diesem Schritt wird durch den Aufruf des Geocodierer-Dienstprogramms eine Geocodierung im Stapelbetrieb ausgeführt. Eine Stapel-Geocodierung wird normalerweise ausgeführt, wenn ein erheblicher Anteil der Tabelle geocodiert oder erneut geocodiert werden muß. 2. Mit diesem Schritt wird die Tabelle OFFICES mit räumlichen Daten ausgefüllt, die in Form einer einer SHAPE-Datei vorliegen. Da die Tabelle OFFICES vorhanden ist und die Schicht OFFICES/ZONE registriert wurde, hängt das Lade-Dienstprogramm die neuen Datensätze an die vorhandene Tabelle an. 3. Mit diesem Schritt wird die Schicht HAZARD_ZONE mit räumlichen Daten ausgefüllt, die in Form einer einer SHAPE-Datei vorliegen. Da die Tabelle und die Schicht nicht vorhanden sind, erstellt das Lade-Dienstprogramm die Tabelle und registriert die Schicht, bevor die Daten geladen werden.
Räumliche Indizes aktivieren	1. Aktivieren des räumlichen Index für die Spalte LOCATION der Tabelle CUSTOMERS (gseEnableIdx) 2. Aktivieren des räumlichen Index für die Spalte ZONE der Tabelle OFFICES (gseEnableIdx) 3. Aktivieren des räumlichen Index für die Spalte LOCATION der Tabelle OFFICES (gseEnableIdx) 4. Aktivieren des räumlichen Index für die Spalte BOUNDRY der Tabelle HAZARD_ZONE (gseEnableIdx)	Mit diesen Schritten wird der räumliche Index für die Tabellen CUSTOMERS, OFFICES und HAZARD_ZONE aktiviert.

Tabelle 5. DB2 Spatial Extender-Beispielprogramm (Forts.)

Schritte des Beispiel-programms	Aktion	Beschreibung
Automatische Geocodierung aktivieren	1. Aktivieren der automatischen Geocodierung für die Spalten LOCATION und ADDRESS der Tabelle CUSTOMERS (gseEnableAutoGC)	Mit diesem Schritt wird der automatische Aufruf des Geocodierers eingeschaltet. Durch die automatische Geocodierung werden die Spalten LOCATION und ADDRESS der Tabelle CUSTOMERS miteinander synchronisiert für spätere Operationen zum Einfügen und Aktualisieren.
Einfügen/Aktualisieren der Tabelle CUSTOMERS	1. Einfügen einiger Datensätze mit einer anderen Straße (gseInsDelUpd) 2. Aktualisieren einiger Datensätze mit einer neuen Adresse (gseInsDelUpd)	Diese Schritte demonstrieren ein Einfügen und Aktualisieren an der Spalte LOCATION der Tabelle CUSTOMERS. Sobald die automatische Geocodierung aktiviert ist, werden Informationen aus der Spalte ADDRESS automatisch geocodiert, wenn sie in die Spalte LOCATION eingefügt bzw. aktualisiert werden. Dieser Prozeß wurde im vorherigen Schritt aktiviert.
Automatische Geocodierung inaktivieren	1. Inaktivieren der automatischen Geocodierung für die Schicht CUSTOMERS (gseDisableAutoGC) 2. Inaktivieren des räumlichen Index für die Schicht CUSTOMERS (gseDisableIdxCustomersLayer)	Mit diesen Schritten wird der automatische Aufruf des Geocodierers und des räumlichen Index in Vorbereitung des nächsten Schritts inaktiviert (der nächste Schritt umfaßt die erneute Geocodierung der gesamten Tabelle CUSTOMERS). Wenn Sie eine große Menge von Geodaten laden, sollten Sie vor dem Laden der Daten den räumlichen Index inaktivieren und ihn nach Abschluß des Ladevorgangs wieder aktivieren.

Tabelle 5. DB2 Spatial Extender-Beispielprogramm (Forts.)

Schritte des Beispiel-programms	Aktion	Beschreibung
Erneut Geocodieren der Tabelle CUSTOMERS	1. Erneutes Geocodieren der Schicht CUSTOMERS mit einer niedrigeren Genauigkeitsstufe – 90% statt 100% (gseRunGC) 2. Erneut Aktivieren des räumlichen Index für die Schicht CUSTOMERS (gseEnableIdx) 3. Erneut Aktivieren der automatischen Geocodierung mit einer niedrigeren Genauigkeitsstufe – 90% statt 100% (gseEnableAutoGC)	Mit diesen Schritten werden der Geocodierer erneut im Stapelbetrieb ausgeführt, die automatische Geocodierung mit einer niedrigeren Genauigkeitsstufe erneut aktiviert und der räumliche Index und die automatische Geocodierung wieder aktiviert. Diese Aktion wird empfohlen, wenn der Administrator für die räumlichen Daten im Geocodierungsprozeß einen hohen Fehleranteil feststellt. Wenn die Genauigkeitsstufe auf 100% eingestellt ist, kann die Geocodierung einer Adresse fehlschlagen, da in den Bezugs-daten keine entsprechende Adresse gefunden wird. Durch die Reduzierung der Genauigkeitsstufe hat der Geocodierer eine bessere Chance, entsprechende Daten zu finden. Nachdem die Tabelle im Stapelbetrieb geocodiert wurde, werden sowohl die automatische Geocodierung als auch der räumliche Index erneut aktiviert, um die schrittweise steigende Wartung des räumlichen Index und der räumlichen Spalte für spätere Einfüge- und Aktualisierungsoperationen zu ermöglichen.
Erstellen einer Sicht und Registrieren ihrer räumlichen Spalten als Sichtschichten	1. Erstellen einer Sicht, HIGHRISK_CUSTOMERS, basierend auf der Verknüpfung der Tabellen CUSTOMERS und HAZARD_ZONE (gseCreateView) 2. Registrieren der räumlichen Spalten der Sicht als Sicht-schichten (gseRegisterLayer)	Mit diesen Schritten werden eine Sicht erstellt und ihre räumlichen Spalten als Sichtschichten registriert.

Tabelle 5. DB2 Spatial Extender-Beispielprogramm (Forts.)

Schritte des Beispiel-programms	Aktion	Beschreibung
Räumliche Analyse ausführen	1. Ermitteln der durchschnittlichen Entfernung von jeder Niederlassung (ST_Within, ST_Distance) 2. Ermitteln des durchschnittlichen Einkommens pro Kunde und der Prämie für jede Niederlassung (ST_Within) 3. Ermitteln der Kunden, die nicht durch eine vorhandene Niederlassung betreut werden (ST_Within) 4. Ermitteln der Gefahrenzonen, die jede Niederlassungszone überlappt (ST_Overlaps) 5. Ermitteln der nächsten Niederlassung von einem bestimmten Kundenstandort aus unter der Annahme, daß sich die Niederlassung im Zentrum der Niederlassungs-zone befindet (ST_Distance, ST_Centroid) 6. Ermitteln der Kunden, deren Standort nahe an der Grenze einer bestimmten Gefahrenzone liegt. (ST_Buffer, ST_Overlaps) 7. Ermitteln der Kunden mit hohem Risiko, die von einer bestimmten Niederlassung betreut werden (Alle diese Schritte verwenden gseRunSpatialQueries)	Mit diesen Schritten wird eine räumliche Analyse durchgeführt mit den räumlichen Prädikaten und Funktionen in der DB2 SQL-Sprache. Der DB2 Query Optimizer nutzt nach Möglichkeit den räumlichen Index mit den räumlichen Spalten zur Verbesserung der Abfrageleistung.
Exportieren der räumlichen Schichten in Dateien	Exportieren der Schicht highRiskCustomers (gseExportShape)	Der Schritt zeigt ein Beispiel zum Exportieren der Ergebnisse Ihrer Abfrage in eine SHAPE-Datei. Das Exportieren der Abfrageergebnisse in ein anderes Dateiformat ermöglicht die Verwendung dieser Informationen durch ein Tool eines anderen Herstellers (z. B. ESRI ArcInfo).

Teil 2. Referenzmaterial

Kapitel 9. Gespeicherte Prozeduren

Dieses Kapitel dokumentiert die gespeicherten Prozeduren, mit deren Hilfe Sie ein geographisches Informationssystem mit dem DB2 Spatial Extender erstellen können. Wenn Sie den DB2 Spatial Extender über das Control Center aktivieren und verwenden, rufen Sie diese gespeicherten Prozeduren implizit auf. Wenn Sie beispielsweise **OK** in einem DB2 Spatial Extender-Fenster anklicken, ruft DB2 die diesem Fenster zugeordnete gespeicherte Prozedur bzw. die Prozeduren auf. Alternativ dazu können Sie die gespeicherten Prozeduren explizit in einem Anwendungsprogramm aufrufen. Es empfiehlt sich, in einem solchen Programm die Kopfdatei db2gse.h anzugeben. Diese Datei enthält die Makrodefinitionen für die Konstanten, die Sie den Parametern der gespeicherten Prozeduren zuordnen. Unter AIX sind diese Definitionen in dem Verzeichnis $DB2INSTANCE/sqllib/include/ gespeichert. Unter Windows NT sind sie im Verzeichnis %DB2PATH%\include\ gespeichert.

Achtung:
Bei allen Zeichenfolgenkonstanten der Eingabeparameter für gespeicherte Prozeduren wird zwischen Groß- und Kleinschreibung unterschieden. Anhand der Tabellen in diesem Kapitel können Sie feststellen, welche Parameter diese Konstanten erfordern.

Bevor Sie eine gespeicherte Prozedur implizit oder explizit aufrufen können, müssen Sie eine Verbindung zu der Datenbank haben, in der der DB2 Spatial Extender installiert ist. Die erste gespeicherte Prozedur, die Sie verwenden, ist db2gse.gse_enable_db. Sie aktiviert die Datenbank für räumliche Operationen. Sie können die weiteren gespeicherten Prozeduren erst verwenden, nachdem die Datenbank aktiviert wurde.

Die Implementierung der gespeicherten Prozeduren sind in der Bibliothek db2gse des DB2 Spatial Extender-Servers archiviert.

In der folgenden Liste können Sie die gespeicherten Prozeduren nach Namen oder nach den ausgeführten Tasks nachschlagen. Die erste Liste zeigt die Namen an:

- „db2gse.gse_enable_idx" auf Seite 83
- „db2gse.gse_enable_sref" auf Seite 85
- „db2gse.gse_export_shape" auf Seite 88
- „db2gse.gse_import_sde" auf Seite 90
- „db2gse.gse_import_shape" auf Seite 92
- „db2gse.gse_register_gc" auf Seite 94
- „db2gse.gse_register_layer" auf Seite 96
- „db2gse.gse_run_gc" auf Seite 102
- „db2gse.gse_unregist_gc" auf Seite 104
- „db2gse.gse_unregist_layer" auf Seite 105

Die nächste Liste zeigt die von den gespeicherten Prozeduren ausgeführten Tasks.

- Einen Index für eine räumliche Spalte erstellen (siehe „db2gse.gse_enable_idx" auf Seite 83).
- Ein räumliches Bezugssystem erstellen (siehe „db2gse.gse_enable_sref" auf Seite 85).
- Einen Geocodierer inaktivieren, so daß er räumliche Spalten nicht mehr automatisch mit den entsprechenden Attributspalten synchronisieren kann (siehe „db2gse.gse_disable_autogc" auf Seite 74).
- Unterstützung für räumliche Operationen in einer Datenbank inaktivieren (siehe „db2gse.gse_disable_db" auf Seite 76).
- Ein räumliches Bezugssystem freigeben (siehe „db2gse.gse_disable_sref" auf Seite 77).
- Eine Datenbank zur Unterstützung räumlicher Operationen aktivieren (siehe „db2gse.gse_enable_db" auf Seite 82).
- Einen Geocodierer aktivieren, so daß er räumliche Spalten automatisch mit den entsprechenden Attributspalten synchronisiert (siehe „db2gse.gse_enable_autogc" auf Seite 78).
- Eine Schicht und ihre zugeordnete Tabelle in eine Formdatei exportieren (siehe „db2gse.gse_export_shape" auf Seite 88).
- Eine Schicht und ihre zugeordnete Tabelle aus einer ESRI_SDE-Übertragungsdatei importieren (siehe „db2gse.gse_import_sde" auf Seite 90).
- Eine Schicht und ihre zugeordnete Tabelle aus einer Formdatei importieren (siehe „db2gse.gse_import_shape" auf Seite 92).

- Einen anderen Geocodierer als den Standard-Codierer registrieren (siehe „db2gse.gse_register_gc" auf Seite 94).
- Eine räumliche Spalte als Schicht registrieren (siehe „db2gse.gse_register_layer" auf Seite 96).
- Einen Geocodierer im Stapelbetrieb ausführen (siehe „db2gse.gse_unregist_gc" auf Seite 104).
- Die Registrierung eines anderen Geocodierers als des Standard-Codierers aufheben (siehe „db2gse.gse_unregist_layer" auf Seite 105).
- Die Registrierung einer Schicht aufheben (siehe „db2gse.gse_unregist_layer" auf Seite 105).

Informationen zu der Reihenfolge, in der Sie diese Tasks ausführen können, finden Sie in „Kapitel 1. Informationen zum DB2 Spatial Extender" auf Seite 3 und „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

db2gse.gse_disable_autogc

Verwenden Sie diese gespeicherte Prozedur zum Freigeben oder zum vorübergehenden Inaktivieren der Auslöser, die für die Synchronisation einer räumlichen Spalte mit den entsprechenden attributiven Spalten sorgen. Es empfiehlt sich beispielsweise, die Auslöser zu inaktivieren, während die Werte in den Attributspalten im Stapelbetrieb geocodiert werden. Weitere Informationen hierzu finden Sie im Abschnitt „Informationen zur Geocodierung" auf Seite 43.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseDisableAutoGc im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine entsprechende Berechtigung haben:

- SYSADM- oder DBADM-Berechtigung für die Datenbank, die die Tabelle enthält, in der die freigegebenen bzw. vorübergehend inaktivierten Auslöser definiert sind.
- Die Berechtigung CONTROL auf diese Tabelle.
- Die Berechtigungen ALTER, SELECT und UPDATE auf diese Tabelle.

Eingabeparameter

Tabelle 6. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_disable_autogc.

Name	Datentyp	Beschreibung
operMode	SMALLINT	Gibt an, ob die Auslöser freigegeben oder vorübergehend inaktiviert werden sollen. Dieser Parameter kann keine Nullwerte enthalten. **Kommentar:** Verwenden Sie zum Freigeben von Auslösern das Makro GSE_AUTOGC_DROP. Verwenden Sie zum vorübergehenden Inaktivieren der Auslöser das Makro GSE_AUTOGC_INVALIDATE. Wenn Sie feststellen wollen, welche Werte diesen Makros zugeordnet sind, schlagen Sie in der Tabelle db2gse.h nach. Unter AIX ist diese Datei in dem Verzeichnis $DB2INSTANCE/sqllib/include/ gespeichert. Unter Windows NT ist sie im Verzeichnis %DB2PATH%\include\ gespeichert.

Tabelle 6. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_disable_autogc. (Forts.)

Name	Datentyp	Beschreibung
layerSchema	VARCHAR(30)	Name des Schemas, zu dem die im Parameter layerTable angegebene Tabelle oder Sicht gehört. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn Sie keinen Wert für den Parameter layerSchema angeben, wird als Standardwert die Benutzer-ID verwendet, unter der die gespeicherte Prozedur db2gse.gse_disable_autogc aufgerufen wurde.
layerTable	VARCHAR(128)	Name der Tabelle, in der die freizugebenden bzw. vorübergehend zu inaktivierenden Auslöser definiert sind. Dieser Parameter kann keine Nullwerte enthalten.
layerColumn	VARCHAR(128)	Name der räumlich aktivierten Spalte, die von den Auslösern verwaltet wird, die Sie freigeben oder vorübergehend inaktivieren wollen. Dieser Parameter kann keine Nullwerte enthalten.

Ausgabeparameter

Tabelle 7. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_disable_autogc.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_disable_db

Verwenden Sie diese gespeicherte Prozedur zum Entfernen von Ressourcen, die dem DB2 Spatial Extender das Speichern räumlicher Daten und die Unterstützung von Operationen mit diesen Daten ermöglichen.

Der Zweck dieser gespeicherten Prozedur ist es, das Lösen von Problemen zu erleichtern, die nach dem Aktivieren der Datenbank für räumliche Operationen auftreten können, aber *vor* dem Hinzufügen von räumlichen Tabellenspalten oder Daten. Es könnte beispielsweise nach dem Aktivieren einer Datenbank für räumliche Operationen entschieden werden, daß statt dessen der DB2 Spatial Extender für eine andere Datenbank verwendet werden soll. Solange keine räumlichen Spalten definiert oder räumliche Daten importiert wurden, können Sie diese gespeicherte Prozedur aufrufen, um alle räumlichen Ressourcen aus der ersten Datenbank zu entfernen.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseDisableDB im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine der Berechtigungen SYSADM oder DBADM für die Datenbank haben, von der die DB2 Spatial Extender-Ressourcen entfernt werden sollen.

Ausgabeparameter

Tabelle 8. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_disable_db.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_disable_sref

Verwenden Sie diese gespeicherte Prozedur zum Freigeben eines räumlichen Bezugssystems. Beim Verarbeiten dieser gespeicherten Prozedur werden Informationen zum räumlichen Bezugssystem aus der Katalogsicht DB2GSE.SPATIAL_REF_SYS entfernt. Informationen zu dieser Sicht finden Sie im Abschnitt „DB2GSE.SPATIAL_REF_SYS" auf Seite 118.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseDisableSref im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Autorisierung

Keine erforderlich.

Eingabeparameter

Tabelle 9. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_disable_sref.

Name	Datentyp	Beschreibung
srId	INTEGER	Numerische Kennung des freizugebenden räumlichen Bezugssystems. Dieser Parameter kann keine Nullwerte enthalten.

Ausgabeparameter

Tabelle 10. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_disable_sref.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

Einschränkung

Bevor Sie ein räumliches Bezugssystem freigeben können, müssen Sie die Registrierung aller Schichten, die es verwenden, aufheben. Wenn solche Schichten nicht registriert werden, wird die Anforderung, das räumliche Bezugssystem freizugeben, zurückgewiesen.

db2gse.gse_enable_autogc

Verwenden Sie diese gespeicherte Prozedur für folgende Aufgaben:

- Erstellen von Auslösern, die für die Synchronisierung einer räumlichen Spalte mit ihren entsprechenden Attributspalten sorgen. Jedes Mal, wenn Werte in die Attributspalten eingefügt bzw. darin aktualisiert werden, ruft ein Auslöser einen registrierten Geocodierer auf, um die eingefügten bzw. aktualisierten Werte zu geocodieren und die resultierenden Daten in der räumlichen Spalten abzulegen.
- Reaktivieren der Auslöser, nachdem sie vorübergehend inaktiviert wurden.
- Festlegen, welche Funktion zum Geocodieren der eingefügten und aktualisierten Werte verwendet wird.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseEnableAutoGC im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben” auf Seite 61.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine entsprechende Berechtigung haben:

- SYSADM- oder DBADM-Berechtigung für die Datenbank, die die Tabelle enthält, in der die von dieser gespeicherten Prozedur erstellten Auslöser definiert sind.
- Die Berechtigung CONTROL auf diese Tabelle.
- Die Berechtigungen ALTER, SELECT und UPDATE auf diese Tabelle.

Eingabeparameter

Tabelle 11. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_enable_autogc.

Name	Datentyp	Beschreibung
operMode	SMALLINT	Dieser Wert gibt an, ob die Auslöser, die die Geocodierung starten, zum ersten Mal erstellt werden oder nach einem vorübergehenden Inaktivieren reaktiviert werden sollen. Dieser Parameter kann keine Nullwerte enthalten. **Kommentar:** Verwenden Sie zum Erstellen der Auslöser das Makro GSE_AUTOGC_CREATE. Verwenden Sie zum Reaktivieren der Auslöser das Makro GSE_AUTOGC_RECREATE. Wenn Sie feststellen wollen, welche Werte diesen Makros zugeordnet sind, schlagen Sie in der Tabelle db2gse.h nach. Unter AIX ist diese Datei in dem Verzeichnis $DB2INSTANCE/sqllib/include/ gespeichert. Unter Windows NT ist sie im Verzeichnis %DB2PATH%\include\ gespeichert.
layerSchema	VARCHAR(30)	Name des Schemas, zu dem die im Parameter layerTable angegebene Tabelle gehört. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn Sie keinen Wert für den Parameter layerSchema angeben, wird als Standardwert die Benutzer-ID verwendet, unter der die gespeicherte Prozedur db2gse.gse_enable_autogc aufgerufen wurde.
layerTable	VARCHAR(128)	Name der Tabelle, mit der die von dieser gespeicherten Prozedur erstellten oder reaktivierten Auslöser arbeiten sollen. Dieser Parameter kann keine Nullwerte enthalten.
layerColumn	VARCHAR(128)	Name der räumlichen Spalte, die von den in dieser gespeicherten Prozedur erstellten bzw. reaktivierten Auslösern verwaltet werden soll. Dieser Parameter kann keine Nullwerte enthalten.

Tabelle 11. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_enable_autogc. (Forts.)

Name	Datentyp	Beschreibung
gcId	INTEGER	Kennung des Geocodierers, der von den in dieser gespeicherten Prozedur erstellten bzw. reaktivierten Einfüge- bzw. Aktualisierungsauslösern gestartet wird. Dieser Parameter kann keine Nullwerte enthalten, wenn der Parameter operMode auf GSE_AUTOGC_CREATE gesetzt wurde. Er kann Nullwerte enthalten, wenn operMode auf GSE_AUTOGC_RECREATE gesetzt ist.
precisionLevel	INTEGER	Die Genauigkeit, mit der die Quellendaten den entsprechenden Bezugsdaten entsprechen müssen, damit der Geocodierer die Quellendaten erfolgreich verarbeiten kann. Dieser Parameter kann keine Nullwerte enthalten, wenn der Parameter operMode auf GSE_AUTOGC_CREATE gesetzt wurde. Er kann Nullwerte enthalten, wenn operMode auf GSE_AUTOGC_RECREATE gesetzt ist. **Kommentar:** Die Genauigkeitsstufe kann von 1 bis 100 Prozent reichen.
vendorSpecific	VARCHAR(256)	Vom Hersteller bereitgestellte technische Informationen, beispielsweise der Pfad und Name einer Datei, über die Parameter eingestellt werden. Dieser Parameter kann keine Nullwerte enthalten, wenn der Parameter operMode auf GSE_AUTOGC_CREATE gesetzt wurde. Er kann Nullwerte enthalten, wenn operMode auf GSE_AUTOGC_RECREATE gesetzt ist.

Ausgabeparameter

Tabelle 12. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_enable_autogc.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

Einschränkungen

- Der Parameter layerColumn muß auf eine Spalte verweisen, die als Tabellenschicht registriert wurde.
- Wenn der Parameter operMode auf GSE_AUTOGC_CREATE gesetzt wurde, müssen Sie dem Parameter gcId die Kennung eines registrierten Geocodierers zuordnen.

db2gse.gse_enable_db

Verwenden Sie diese gespeicherte Prozedur, um einer Datenbank die Ressourcen bereitzustellen, die sie zum Speichern räumlicher Operationen und zum Unterstützen von Operationen benötigt. Diese Ressourcen umfassen räumliche Datentypen, einen Typ eines räumlichen Index, Katalogtabellen und -sichten, bereitgestellte Funktionen und andere gespeicherte Prozeduren.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseEnableDB im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben” auf Seite 61.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine der Berechtigungen SYSADM oder DBADM für die zu aktivierende Datenbank haben.

Ausgabeparameter

Tabelle 13. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_enable_db.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_enable_idx

Verwenden Sie diese gespeicherte Prozedur zum Erstellen eines Index für eine räumliche Spalte.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseEnableIdx im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine der folgenden Berechtigungen haben:

- SYSADM- oder DBADM-Berechtigung für die Datenbank, die die Tabelle enthält, für die der aktivierte Index verwendet werden soll.
- Die Berechtigung CONTROL oder INDEX auf diese Tabelle.

Eingabeparameter

Tabelle 14. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_enable_idx.

Name	Datentyp	Beschreibung
layerSchema	VARCHAR(30)	Name des Schemas, zu dem die im Parameter layerTable angegebene Tabelle gehört. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn Sie keinen Wert für den Parameter layerSchema angeben, wird als Standardwert die Benutzer-ID verwendet, unter der die gespeicherte Prozedur db2gse.gse_enable_idx aufgerufen wurde.
layerTable	VARCHAR(128)	Name der Tabelle, in der der zu erstellende Index definiert werden soll. Dieser Parameter kann keine Nullwerte enthalten.
layerColumn	VARCHAR(128)	Name der räumlich aktivierten Spalte, die mit Hilfe des zu erstellenden Index durchsucht werden soll. Dieser Parameter kann keine Nullwerte enthalten.

Tabelle 14. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_enable_idx. (Forts.)

Name	Datentyp	Beschreibung
indexName	VARCHAR(128)	Name des zu erstellenden Index. Dieser Parameter kann keine Nullwerte enthalten. **Kommentar:** Geben Sie keinen Schemanamen an. Der DB2 Spatial Extender ordnet den Index automatisch dem Schema zu, auf das vom Parameter layerSchema verwiesen wird.
gridSize1	DOUBLE	Diese Zahl gibt an, wie die Unterteilung des feinsten Gitters sein soll. Dieser Parameter kann keine Nullwerte enthalten.
gridSize2	DOUBLE	Diese Zahl gibt an, (1) daß für diesen Index kein zweites Gitter vorhanden sein soll oder (2) wie die Unterteilung des zweiten Gitters sein soll. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn kein zweites Gitter vorhanden sein soll, geben Sie 0 an. Soll ein zweites Gitter verwendet werden, muß es feiner sein als das mit gridSize1 angegebene Gitter.
gridSize3	DOUBLE	Diese Zahl gibt an, (1) daß für diesen Index kein drittes Gitter vorhanden sein soll oder (2) wie die Unterteilung des dritten Gitters sein soll.Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn kein drittes Gitter vorhanden sein soll, geben Sie 0 an. Soll ein drittes Gitter verwendet werden, muß es feiner sein als das mit gridSize2 angegebene Gitter.

Ausgabeparameter

Tabelle 15. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_enable_idx.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_enable_sref

Geben Sie über diese gespeicherte Prozedur an, wie negative und Dezimalzahlen in einem spezifischen Koordinatensystem in positive ganze Zahlen umgewandelt werden sollen, so daß sie vom DB2 Spatial Extender gespeichert werden können. Ihre Angaben werden als *räumliches Bezugssystem* bezeichnet. Beim Verarbeiten dieser gespeicherten Prozedur werden der Katalogsicht DB2GSE.SPATIAL_REF_SYS Informationen zum räumlichen Bezugssystem hinzugefügt. Informationen zu dieser Sicht finden Sie im Abschnitt „DB2GSE.SPATIAL_REF_SYS" auf Seite 118.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseEnableSref im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Autorisierung

Keine erforderlich.

Eingabeparameter

Tabelle 16. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_enable_sref.

Name	Datentyp	Beschreibung
srId	INTEGER	Eine numerische Kennung für dieses räumliche Bezugssystem. Dieser Parameter kann keine Nullwerte enthalten. **Kommentar:** Diese Kennung muß innerhalb der räumlich aktivierten Datenbank eindeutig sein.
srName	VARCHAR(64)	Kurzbeschreibung des räumlichen Bezugssystems. Dieser Parameter kann keine Nullwerte enthalten. **Kommentar:** Diese Beschreibung muß innerhalb der räumlich aktivierten Datenbank eindeutig sein.
falsex	DOUBLE	Eine Zahl, die von einem negativen X-Koordinatenwert abgezogen wird, um auf diese Weise eine nicht negative Zahl zu erhalten (eine positive Zahl oder Null). Dieser Parameter kann keine Nullwerte enthalten.

Tabelle 16. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_enable_sref. (Forts.)

Name	Datentyp	Beschreibung
falsey	DOUBLE	Eine Zahl, die von einem negativen Y-Koordinatenwert abgezogen wird, um auf diese Weise eine nicht negative Zahl zu erhalten (eine positive Zahl oder Null). Dieser Parameter kann keine Nullwerte enthalten.
xyunits	DOUBLE	Eine Zahl, die mit einer dezimalen X-Koordinate oder einer dezimalen Y-Koordinate multipliziert wird, um eine ganze Zahl zu erhalten, die als 32-Bit-Datenelement gespeichert werden kann. Dieser Parameter kann keine Nullwerte enthalten.
falsez	DOUBLE	Eine Zahl, die von einem negativen Z-Koordinatenwert abgezogen wird, um auf diese Weise eine nicht negative Zahl zu erhalten (eine positive Zahl oder Null). Dieser Parameter kann keine Nullwerte enthalten.
zunits	DOUBLE	Eine Zahl, die mit einer dezimalen Z-Koordinate multipliziert wird, um eine ganze Zahl zu erhalten, die als 32-Bit-Datenelement gespeichert werden kann. Dieser Parameter kann keine Nullwerte enthalten.
falsem	DOUBLE	Eine Zahl, die von einem negativen Maß abgezogen wird, um auf diese Weise eine nicht negative Zahl zu erhalten (eine positive Zahl oder Null). Dieser Parameter kann keine Nullwerte enthalten.
munits	DOUBLE	Eine Zahl, die mit einem dezimalen Maß multipliziert wird, um eine ganze Zahl zu erhalten, die als 32-Bit-Datenelement gespeichert werden kann. Dieser Parameter kann keine Nullwerte enthalten.

Tabelle 16. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_enable_sref. (Forts.)

Name	Datentyp	Beschreibung
scId	INTEGER	Numerische Kennung des Koordinatensystems, aus dem das räumliche Bezugssystem abgeleitet wird. Die numerische Kennung des Koordinatensystems können Sie in der Katalogsicht DB2GSE.COORD_REF_SYS im Abschnitt „DB2GSE.COORD_REF_SYS" auf Seite 115 ermitteln. Dieser Parameter kann keine Nullwerte enthalten.

Ausgabeparameter

Tabelle 17. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_enable_sref.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_export_shape

Verwenden Sie diese gespeicherte Prozedur zum Exportieren einer Schicht und ihrer zugeordneten Tabelle in eine Formdatei oder zum Erstellen einer neuen Formdatei und zum Exportieren einer Schicht und ihrer zugeordneten Tabelle in diese neue Datei.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseExportShape im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß die Berechtigung SELECT auf die zu exportierende Tabelle haben.

Eingabeparameter

Tabelle 18. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_export_shape.

Name	Datentyp	Beschreibung
layerSchema	VARCHAR(30)	Name des Schemas, zu dem die im Parameter layerTable angegebene Tabelle gehört. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn Sie keinen Wert für den Parameter layerSchema angeben, wird als Standardwert die Benutzer-ID verwendet, unter der die gespeicherte Prozedur db2gse.gse_export_shape aufgerufen wurde.
layerTable	VARCHAR(128)	Name der zu exportierenden Tabelle. Dieser Parameter kann keine Nullwerte enthalten.
layerColumn	VARCHAR(30)	Name der Spalte, die als die zu exportierende Schicht registriert wurde. Dieser Parameter kann keine Nullwerte enthalten.
fileName	VARCHAR(128)	Name der Formdatei, in die die angegebene Schicht exportiert werden soll. Dieser Parameter kann keine Nullwerte enthalten.

Tabelle 18. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_export_shape. (Forts.)

Name	Datentyp	Beschreibung
whereClause	VARCHAR(1024)	Der Hauptteil der SQL WHERE-Klausel. Er definiert eine Einschränkung zu der Gruppe der zu geocodierenden Datensätze. Die Klausel kann auf eine Attributspalte in der zu exportierenden Tabelle verweisen. Dieser Parameter kann Nullwerte enthalten.

Ausgabeparameter

Tabelle 19. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_export_shape.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

Einschränkung

Sie können immer nur eine Schicht auf einmal exportieren.

db2gse.gse_import_sde

Verwenden Sie diese gespeicherte Prozedur zum Importieren einer SDE-Übertragungsdatei in eine Datenbank, die für räumliche Operationen aktiviert wurde. Die gespeicherte Prozedur kann auf eine von zwei verschiedenen Arten arbeiten:

- Wenn die SDE-Übertragungsdatei für eine vorhandene Tabelle bestimmt ist, die eine registrierte Schichtspalte enthält, lädt der DB2 Spatial Extender die Tabelle mit den Daten der Datei.
- Andernfalls erstellt der DB2 Spatial Extender eine Tabelle mit einer räumlichen Spalte, registriert diese Spalte als Schicht und lädt die Schicht sowie die anderen Spalten der Tabelle mit den Daten aus der Datei.

Das in der SDE-Übertragungsdatei angegebene räumliche Bezugssystem wird mit den räumlichen Bezugssystemen verglichen, die für den DB2 Spatial Extender registriert wurden. Wenn das angegebene System einem registrierten System entspricht, werden die negativen und Dezimalwerte in der Übertragungsdatei nach dem Laden auf die durch das registrierte System vorgegebene Weise geändert. Entspricht das angegebene System keinem der registrierten Systeme, erstellt der DB2 Spatial Extender ein neues räumliches Bezugssystem zur Definition der Änderungsvorschriften.

Autorisierung

Wenn Sie Daten in eine vorhandene Tabelle importieren, muß die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, eine der folgenden Berechtigungen haben:

- SYSADM- oder DBADM-Berechtigung für die Datenbank, die die Tabelle enthält, in die die Daten importiert werden sollen.
- Die Berechtigung CONTROL auf diese Tabelle.

Wenn die Tabelle, in die Daten importiert werden sollen, zunächst erstellt werden muß, muß die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, eine der folgenden Berechtigungen haben:

- SYSADM- oder DBADM-Berechtigung für die Datenbank, die die zu erstellende Tabelle enthält.

Eingabeparameter

Tabelle 20. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_import_sde.

Name	Datentyp	Beschreibung
layerSchema	VARCHAR(30)	Name des Schemas, zu dem die im Parameter layerTable angegebene Tabelle oder Sicht gehört. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn Sie keinen Wert für den Parameter layerSchema angeben, wird als Standardwert die Benutzer-ID verwendet, unter der die gespeicherte Prozedur db2gse.gse_import_sde aufgerufen wurde.
layerTable	VARCHAR(128)	Name der Tabelle, in die die Daten aus der SDE-Übertragungsdatei geladen werden sollen. Dieser Parameter kann keine Nullwerte enthalten.
layerColumn	VARCHAR(30)	Name der Spalte, die als die Schicht registriert wurde, in die die räumlichen Daten aus der SDE-Übertragungsdatei geladen werden sollen. Dieser Parameter kann keine Nullwerte enthalten.
fileName	VARCHAR(128)	Name der zu importierenden SDE-Übertragungsdatei. Dieser Parameter kann keine Nullwerte enthalten.
commitScope	INTEGER	Anzahl der Datensätze pro Prüfpunkt. Dieser Parameter kann Nullwerte enthalten.

Ausgabeparameter

Tabelle 21. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_import_sde.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_import_shape

Verwenden Sie diese gespeicherte Prozedur zum Importieren einer Formdatei in eine Datenbank, die für räumliche Operationen aktiviert wurde. Die gespeicherte Prozedur kann auf eine von zwei verschiedenen Arten arbeiten:

- Wenn die Formdatei für eine vorhandene Tabelle bestimmt ist, die eine registrierte Schichtspalte enthält, lädt der DB2 Spatial Extender die Tabelle mit den Daten der Datei.
- Andernfalls erstellt der DB2 Spatial Extender eine Tabelle mit einer räumlichen Spalte, registriert diese Spalte als Schicht und lädt die Schicht sowie die anderen Spalten der Tabelle mit den Daten aus der Datei.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseImportShape im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine der folgenden Berechtigungen haben:

- SYSADM- oder DBADM-Berechtigung für die Datenbank, die die Tabelle enthält, in die die importierten Formdaten geladen werden sollen.
- Die Berechtigung CONTROL auf diese Tabelle.

Eingabeparameter

Tabelle 22. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_import_shape.

Name	Datentyp	Beschreibung
layerSchema	VARCHAR(30)	Name des Schemas, zu dem die im Parameter layerTable angegebene Tabelle oder Sicht gehört. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn Sie keinen Wert für den Parameter layerSchema angeben, wird als Standardwert die Benutzer-ID verwendet, unter der die gespeicherte Prozedur db2gse.gse_import_shape aufgerufen wurde.
layerTable	VARCHAR(128)	Name der Tabelle, in die die importierte Formdatei geladen werden sollen. Dieser Parameter kann keine Nullwerte enthalten.

Tabelle 22. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_import_shape. (Forts.)

Name	Datentyp	Beschreibung
layerColumn	VARCHAR(30)	Name der Spalte, die als die Schicht registriert wurde, in die die Formdate geladen werden sollen. Dieser Parameter kann keine Nullwerte enthalten.
fileName	VARCHAR(128)	Name der zu importierenden Formdatei. Dieser Parameter kann keine Nullwerte enthalten.
exceptionFile	VARCHAR(128)	Pfad und Name der Datei, in der die Formen gespeichert werden sollen, die nicht importiert werden konnten. Diese ist eine neue Datei, die bei der Ausführung der gespeicherten Prozedur db2gse.gse_import_shape erstellt wird. Dieser Parameter kann keine Nullwerte enthalten.
srId	INTEGER	Kennung des für die Schicht zu verwendenden räumlichen Bezugssystems, in das die Formdaten geladen werden sollen. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn diese Kennung nicht angegeben ist, wird die interne Umwandlung auf die maximal mögliche Auflösung für die Formdatei eingestellt.
commitScope	INTEGER	Anzahl der Datensätze pro Prüfpunkt. Dieser Parameter kann Nullwerte enthalten.

Ausgabeparameter

Tabelle 23. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_import_shape.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_register_gc

Verwenden Sie diese gespeicherte Prozedur zum Registrieren eines anderen Geocodierers als des Standard-Codierers. Wenn Sie feststellen wollen, ob bereits ein Geocodierer registriert wurde, schlagen Sie in der Katalogsicht DB2GSE.SPATIAL_GEOCODER (siehe Abschnitt „DB2GSE.SPATIAL_GEOCODER" auf Seite 117) nach.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine der Berechtigungen SYSADM oder DBADM für die zu aktivierende Datenbank haben.

Eingabeparameter

Tabelle 24. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_register_gc.

Name	Datentyp	Beschreibung
gcId	INTEGER	Numerische Kennung des zu registrierenden Geocodierers. Dieser Parameter kann keine Nullwerte enthalten. **Kommentar:** Diese Kennung muß innerhalb der Datenbank eindeutig sein.
gcName	VARCHAR(64)	Kurzbeschreibung des zu registrierenden Geocodierers. Dieser Parameter kann keine Nullwerte enthalten. **Kommentar:** Diese Beschreibung muß eine innerhalb der Datenbank eindeutige Zeichenfolge sein.
vendorName	VARCHAR(64)	Name des Herstellers, der den zu registrierenden Geocodierer bereitgestellt hat. Dieser Parameter kann keine Nullwerte enthalten.
primaryUDF	VARCHAR(256)	Vollständig qualifizierter Name des zu registrierenden Geocodierers. Dieser Parameter kann keine Nullwerte enthalten.

Tabelle 24. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_register_gc. (Forts.)

Name	Datentyp	Beschreibung
precisionLevel	INTEGER	Die Genauigkeit, mit der die Quellendaten den entsprechenden Bezugsdaten entsprechen müssen, damit der Geocodierer die Quellendaten erfolgreich verarbeiten kann. Dieser Parameter kann keine Nullwerte enthalten. **Kommentar:** Die Genauigkeitsstufe kann von 1 bis 100 Prozent reichen.
vendorSpecific	VARCHAR(256)	Vom Hersteller bereitgestellte technische Informationen, beispielsweise der Pfad und Name einer Datei, über die Parameter eingestellt werden. Dieser Parameter kann Nullwerte enthalten.
geoArea	VARCHAR(256)	Der zu geocodierende geographische Bereich. Dieser Parameter kann Nullwerte enthalten.
Beschreibung	VARCHAR(256)	Anmerkungen des Herstellers. Dieser Parameter kann Nullwerte enthalten.

Ausgabeparameter

Tabelle 25. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_register_gc.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_register_layer

Verwenden Sie diese gespeicherte Prozedur zum Registrieren einer räumlichen Spalte als Schicht. Beim Verarbeiten dieser gespeicherten Prozedur werden der Katalogsicht DB2GSE.GEOMETRY_COLUMNS Informationen zu der zu registrierenden Schicht hinzugefügt. Informationen zu dieser Sicht finden Sie im Abschnitt „DB2GSE.GEOMETRY_COLUMNS" auf Seite 116.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseRegisterLayer im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine der folgenden Berechtigungen haben:

- Für eine Tabellenspalte:
  - SYSADM- oder DBADM-Berechtigung für die Datenbank, die die Tabelle enthält, zu der diese Schicht gehört.
  - Die Berechtigung CONTROL oder ALTER auf diese Tabelle.
- Für eine Sichtstufe:
  - Die Berechtigung SELECT für die Basistabelle oder -tabellen, die (1) die zu geocodierenden Adreßdaten für diese Schicht enthalten und (2) die räumlichen Daten enthalten, die aus der Geocodierung resultieren.

Eingabeparameter

Tabelle 26. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_register_layer.

Name	Datentyp	Beschreibung
layerSchema	INTEGER(30)	Name des Schemas, zu dem die im Parameter layerTable angegebene Tabelle oder Sicht gehört. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn Sie keinen Wert für den Parameter layerSchema angeben, wird als Standardwert die Benutzer-ID verwendet, unter der die gespeicherte Prozedur db2gse.gse_register_layer aufgerufen wurde.
layerTable	VARCHAR(128)	Der Name der Tabelle, die die Spalte enthält, die als Schicht registriert werden soll. Dieser Parameter kann keine Nullwerte enthalten.

Tabelle 26. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_register_layer. (Forts.)

Name	Datentyp	Beschreibung
layerColumn	VARCHAR(128)	Name der als Schicht zu registrierenden Spalte. Wenn die Spalte nicht vorhanden ist, erstellt der DB2 Spatial Extender sie. Dieser Parameter kann keine Nullwerte enthalten.
layerTypeName	VARCHAR(64)	Datentyp der als Schicht zu registrierenden Spalte. Sie müssen den Datentyp in Großbuchstaben eingeben, z. B.: `ST_POINT` Dieser Parameter kann keine Nullwerte enthalten, wenn die Spalte eine Tabellenspalte ist, die bei der Verarbeitung dieser gespeicherten Prozedur erstellt werden soll. Andernfalls, sofern die Spalte eine vorhandene Spalte in einer Tabelle oder Sicht ist, kann dieser Parameter Nullwerte enthalten.
srId	INTEGER	Kennung des für diese Schicht verwendeten räumlichen Bezugssystems. Dieser Parameter kann für eine Tabellenschicht keine Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtschicht.
geoSchema	VARCHAR(30)	Wird beim Registrieren einer Sichtspalte als Schicht verwendet. Der Parameter geoSchema ist das Schema der Tabelle, das der Sicht zugrunde liegt, zu der die Spalte gehört. Dieser Parameter kann beim Registrieren einer Sichtspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Tabellenspalte als Schicht. **Kommentar:** Wenn Sie keinen Wert für den Parameter geoSchema angeben, wird als Standardwert der Wert des Parameters layerSchema verwendet.

Tabelle 26. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_register_layer. (Forts.)

Name	Datentyp	Beschreibung
geoTable	VARCHAR(128)	Wird beim Registrieren einer Sichtspalte als Schicht verwendet. Der Parameter geoTable ist der Name der Tabelle, die der Sicht zugrunde liegt, zu der die Spalte gehört. Dieser Parameter kann beim Registrieren einer Sichtspalte als Schicht keine Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Tabellenspalte als Schicht.
geoColumn	VARCHAR(128)	Wird beim Registrieren einer Sichtspalte als Schicht verwendet. Der Parameter geoColumn ist der Name der Tabellenspalte, die dieser Sichtspalte zugrunde liegt. Dieser Parameter kann beim Registrieren einer Sichtspalte als Schicht keine Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Tabellenspalte als Schicht.
nAttributes	SMALLINT	Anzahl der Spalten, die die für diese Schicht zu geocodierenden Quellendaten enthalten. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht.
attr1Name	VARCHAR(128)	Name der ersten Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Wenn Sie den Standard-Geocodierer verwenden wollen, müssen Sie die Straßennamen in der Spalte attr1Name speichern.

Tabelle 26. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_register_layer. (Forts.)

Name	Datentyp	Beschreibung
attr2Name	VARCHAR(128)	Name der zweiten Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Wenn Sie den Standard-Geocodierer verwenden wollen, müssen Sie die Ortsnamen in der Spalte attr2Name speichern.
attr3Name	VARCHAR(128)	Name der dritten Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Wenn Sie den Standard-Geocodierer verwenden wollen, müssen Sie die Namen bzw. Abkürzungen der Bundesstaaten in der Spalte attr3Name speichern.
attr4Name	VARCHAR(128)	Name der vierten Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Wenn Sie den Standard-Geocodierer verwenden wollen, müssen Sie die Postleitzahl in der Spalte attr4Name speichern.

Tabelle 26. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_register_layer. (Forts.)

Name	Datentyp	Beschreibung
attr5Name	VARCHAR(128)	Name der fünften Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Der Standard-Geocodierer ignoriert die Spalte Attr5Name.
attr6Name	VARCHAR(128)	Name der sechsten Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Der Standard-Geocodierer ignoriert die Spalte Attr6Name.
attr7Name	VARCHAR(128)	Name der siebten Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Der Standard-Geocodierer ignoriert die Spalte Attr7Name.
attr8Name	VARCHAR(128)	Name der achten Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Der Standard-Geocodierer ignoriert die Spalte Attr8Name.

Tabelle 26. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_register_layer. (Forts.)

Name	Datentyp	Beschreibung
attr9Name	VARCHAR(128)	Name der neunten Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Der Standard-Geocodierer ignoriert die Spalte Attr9Name.
attr10Name	VARCHAR(128)	Name der zehnten Spalte, die zu geocodierende Quellendaten für diese Schicht enthält. Dieser Parameter kann beim Registrieren einer Tabellenspalte als Schicht Nullwerte enthalten. Der DB2 Spatial Extender ignoriert diesen Parameter beim Registrieren einer Sichtspalte als Schicht. Der Standard-Geocodierer ignoriert die Spalte Attr10Name.

Ausgabeparameter

Tabelle 27. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_register_layer.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

Einschränkungen

- Wenn Sie eine Sichtspalte als Schicht registrieren, muß sie auf einer Tabellenspalte basieren, die bereits als Schicht registriert wurde.
- Die zu geocodierenden Daten für die zu registrierende Schicht können nicht auf mehr als zehn Attributspalten verteilt sein.

db2gse.gse_run_gc

Verwenden Sie diese gespeicherte Prozedur zum Ausführen eines Geocodierers im Stapelbetrieb. Informationen zu dieser Task finden Sie im Abschnitt „Den Geocodierer im Stapelbetrieb ausführen" auf Seite 46.

Ein Beispiel für den Code zum Aufrufen dieser gespeicherten Prozedur finden Sie in der C-Funktion gseRunGC im Beispielprogramm. Informationen zu diesem Programm finden Sie in „Kapitel 8. Anwendungen für DB2 Spatial Extender schreiben" auf Seite 61.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine der folgenden Berechtigungen haben:

- SYSADM- oder DBADM-Berechtigung für die Datenbank, die die Tabelle enthält, mit der der angegebene Geocodierer arbeiten soll.
- Die Berechtigung CONTROL oder UPDATE auf diese Tabelle.

Eingabeparameter

Tabelle 28. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_run_gc.

Name	Datentyp	Beschreibung
layerSchema	VARCHAR(30)	Name des Schemas, zu dem die im Parameter layerTable angegebene Tabelle oder Sicht gehört. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn Sie keinen Wert für den Parameter layerSchema angeben, wird als Standardwert die Benutzer-ID verwendet, unter der die gespeicherte Prozedur db2gse.gse_run_gc aufgerufen wurde.
layerTable	VARCHAR(128)	Der Name der Tabelle, die die Spalte enthält, in die die geocodierten Daten eingefügt werden sollen. Dieser Parameter kann keine Nullwerte enthalten.
layerColumn	VARCHAR(128)	Der Name der Spalte, in die die geocodierten Daten eingefügt werden sollen. Dieser Parameter kann keine Nullwerte enthalten.

Tabelle 28. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_run_gc. (Forts.)

Name	Datentyp	Beschreibung
gcId	INTEGER	Kennung des auszuführenden Geocodierers. Dieser Parameter kann Nullwerte enthalten. Zum Ermitteln der Kennungen der registrierten Geocodierer schlagen Sie in der Katalogsicht DB2GSE.SPATIAL_GEOCODER nach.
precisionLevel	INTEGER	Die Genauigkeit, mit der die Quellendaten den entsprechenden Bezugsdaten entsprechen müssen, damit der Geocodierer die Quellendaten erfolgreich verarbeiten kann. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Die Genauigkeitsstufe kann von 1 bis 100 Prozent reichen.
vendorSpecific	VARCHAR(256)	Vom Hersteller bereitgestellte technische Informationen, beispielsweise der Pfad und Name einer Datei, über die Parameter eingestellt werden. Dieser Parameter kann Nullwerte enthalten.
whereClause	VARCHAR(256)	Der Hauptteil der WHERE-Klausel. Er definiert eine Einschränkung zu der Gruppe der zu geocodierenden Datensätze. Die Klausel kann auf eine Attributspalte in der Tabelle, mit der der Geocodierer arbeiten soll, verweisen. Dieser Parameter kann Nullwerte enthalten.
commitScope	INTEGER	Anzahl der Datensätze pro Prüfpunkt. Dieser Parameter kann Nullwerte enthalten.

Ausgabeparameter

Tabelle 29. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_run_gc.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_unregist_gc

Verwenden Sie diese gespeicherte Prozedur, um die Registrierung eines anderen Geocodierers als des Standard-Codierers aufzuheben.

Wenn Sie Informationen über den Geocodierer suchen, dessen Registrierung aufgehoben werden soll, schlagen in der Katalogsicht DB2GSE.SPATIAL_GEOCODER nach; siehe hierzu den Abschnitt „DB2GSE.SPATIAL_GEOCODER" auf Seite 117.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine der Berechtigungen SYSADM oder DBADM für die Datenbank haben, die den Geocodierer enthält, dessen Berechtigung aufgehoben werden soll.

Eingabeparameter

Tabelle 30. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_unregist_gc.

Name	Datentyp	Beschreibung
gcId	INTEGER	Die Kennung des Geocodierers, dessen Registrierung aufgehoben werden soll. Dieser Parameter kann keine Nullwerte enthalten.

Ausgabeparameter

Tabelle 31. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_unregist_gc.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

db2gse.gse_unregist_layer

Verwenden Sie diese gespeicherte Prozedur, um die Registrierung einer Schicht aufzuheben. Die gespeicherte Prozedur führt hierzu folgende Aktionen aus:

- Entfernen der Definition der Schicht aus den DB2 Spatial Extender-Katalogtabellen.
- Löschen der Prüfeinschränkung, die der DB2 Spatial Extender auf die Basistabelle dieser Schicht plaziert hat, um sicherzustellen, daß die räumlichen Daten der Schicht den Anforderungen des räumlichen Bezugssystems der Schicht entsprechen.
- Freigeben der Auslöser, mit denen die räumliche Spalte beim Hinzufügen, Ändern oder Entfernen von Adreßdaten aktualisiert wird.

Beim Verarbeiten dieser gespeicherten Prozedur werden Informationen über die Schicht aus der Meta-Sicht DB2GSE.GEOMETRY_COLUMNS entfernt. Informationen zu dieser Sicht finden Sie im Abschnitt „DB2GSE.GEOMETRY_COLUMNS" auf Seite 116.

Autorisierung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muß eine der folgenden Berechtigungen haben:

- Für eine Tabellenspalte:
  - SYSADM- oder DBADM-Berechtigung für die Datenbank, die die Basistabelle zu dieser Schicht enthält.
  - Die Berechtigung CONTROL oder ALTER auf diese Tabelle.
- Für eine Sichtstufe:
  - Die Berechtigung SELECT für die Basistabelle oder -tabellen, die (1) die zu geocodierenden Adreßdaten für diese Schicht enthalten und (2) die räumlichen Daten enthalten, die aus der Geocodierung resultieren.

Eingabeparameter

Tabelle 32. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_unregist_layer.

Name	Datentyp	Beschreibung
layerSchema	VARCHAR(30)	Name des Schemas, zu dem die im Parameter layerTable angegebene Tabelle gehört. Dieser Parameter kann Nullwerte enthalten. **Kommentar:** Wenn Sie keinen Wert für den Parameter layerSchema angeben, wird als Standardwert die Benutzer-ID verwendet, unter der die gespeicherte Prozedur db2gse.gse_unregister_layer aufgerufen wurde.

Tabelle 32. Eingabeparameter für die gespeicherte Prozedur db2gse.gse_unregist_layer. (Forts.)

Name	Datentyp	Beschreibung
layerTable	VARCHAR(128)	Der Name der Tabelle, die die im Parameter layerColumn angegebene Spalte enthält. Dieser Parameter kann keine Nullwerte enthalten.
layerColumn	VARCHAR(128)	Name der räumlichen Spalte, die als die Schicht definiert wurde, deren Registrierung aufgehoben werden soll. Dieser Parameter kann keine Nullwerte enthalten.

Ausgabeparameter

Tabelle 33. Ausgabeparameter für die gespeicherte Prozedur db2gse.gse_unregist_layer.

Name	Datentyp	Beschreibung
msgCode	INTEGER	Code, der den Nachrichten zugeordnet ist, die der Aufrufer dieser gespeicherten Prozedur zurückgeben kann.
Reserviert	VARCHAR(1024)	Vollständige Fehlernachricht, wie auf dem DB2 Spatial Extender-Server konstruiert.

Einschränkung

Wenn eine als Sichtstufe definierte Sichtspalte auf einer Tabellenspalte basiert, die als Tabellenschicht definiert wurde, können Sie die Registrierung dieser Tabellenschicht nicht aufheben, bis die Registrierung der Sichtschicht aufgehoben wurde.

Kapitel 10. Nachrichten

Dieses Kapitel dokumentiert Nachrichten, die der DB2 Spatial Extender an die Benutzer zurückgibt. Jede Nachricht hat eine Kennung. Kennungen, die auf E enden, weisen auf Fehlernachrichten hin; Kennungen auf W stehen für Warnungen, und Kennungen auf I stehen für allgemeine Informationen.

DBA7200E More than 10 columns are selected as input to a geocoder

Erläuterung: Es können bis zu 10 Spalten als Eingabe für einen Geocodierer ausgewählt werden.

Benutzerantwort: Verschieben Sie Spaltennamen aus dem Feld ″Selected columns″ in das Feld ″Available columns″, bis das Feld ″Selected columns″ maximal zehn Namen enthält.

DBA7201E The database is not enabled for Spatial Extender operation.

Erläuterung: Die Datenbank muß für Spatial Extender aktiviert werden, bevor Sie mit Spatial Extender arbeiten können.

Benutzerantwort: Klicken Sie mit der rechten Maustaste die Datenbank an, und wählen Sie Spatial Extender —> Enable in den Menüs aus.

GSE0000I The operation is completed successfully.

GSE0001E Spatial Extender could not perform the requested operation (“<operation-name>”) under user ID “<benutzer-ID>”.

Erläuterung: Sie haben diese Operation mit einer Benutzer-ID angefordert, die keine ausreichende Berechtigung für diese Aktion hat.

Benutzerantwort: Schlagen Sie in der Dokumentation nach, um festzustellen, welche Berechtigung erforderlich ist, und fordern Sie diese Berechtigung bei Ihrem Spatial Extender Administrator an.

GSE0002E “<wert>” is not a valid value for the “<argumentname>” argument.

Erläuterung: Der eingegebene Wert war falsch oder falsch geschrieben.

Benutzerantwort: Schlagen Sie in der Dokumentation nach oder fragen Sie Ihren Spatial Extender Administrator, welcher Wert bzw. welcher Wertebereich angegeben werden muß.

GSE0003E Spatial Extender could not perform the requested operation because argument “<argumentname>” was not specified.

Erläuterung: Sie haben ein für diese Operation erforderliches Argument nicht angegeben.

Benutzerantwort: Geben Sie das Argument “<argument-name>” mit dem gewünschten Wert an, und fordern Sie die Operation erneut an.

GSE0004W The argument “<argumentname>” was not evaluated.

Erläuterung: Die angeforderte Operation verwendet das Argument “<argumentname>” nicht.

Benutzerantwort: Keine erforderlich.

GSE0005E Spatial Extender could not process your request to create an object named “<objektname>”.

Erläuterung: Das Objekt “<objektname>” ist bereits vorhanden, oder Sie haben nicht die erforderliche Berechtigung, um es zu erstellen. Das Objekt kann eine Tabelle, eine Spalte, ein Auslöser, ein Index oder ein anderes Objekt sein.

Benutzerantwort: Wenn "<objektname>" das gewünschte Objekt ist, brauchen Sie keine Aktion auszuführen. Geben Sie andernfalls den Namen richtig ein, und stellen Sie sicher, daß Sie die Berechtigung zum Erstellen des Objekts haben.

GSE0006E Spatial Extender could not perform the requested operation on an enabled or registered object named "<objektname>".

Erläuterung: Das Objekt "<objektname>" wurde bereits aktiviert bzw. registriert, oder es ist bereits vorhanden. Das Objekt kann eine Schicht, ein Index, ein räumliches Bezugssystem, ein Koordinatensystem oder ein anderes Objekt sein.

Benutzerantwort: Stellen Sie sicher, daß das Objekt "<objektname>" vorhanden ist, und geben Sie Ihre Anforderung erneut ein.

GSE0007E Spatial Extender could not perform the requested operation on "<objektname>", an object that has not yet been enabled or registered.

Erläuterung: Das Objekt "<objektname>" wurde nicht aktiviert oder registriert. Das Objekt kann eine Schicht, ein Index, ein räumliches Bezugssystem, ein räumliches Koordinatensystem, ein Geocodierer oder ein anderes Objekt sein.

Benutzerantwort: Aktivieren oder registrieren Sie das Objekt "<objektname>". Übergeben Sie Ihre Anforderung anschließend erneut.

GSE0008E An unexpected SQL error ("<sql-fehlernachricht>") has occurred.

Benutzerantwort: Schlagen Sie die ausführliche Nachricht zum SQLCODE in der SQL-Fehlernachricht "<sql-fehlernachricht>" nach. Wenden Sie sich ggf. an Ihren IBM Ansprechpartner.

GSE0009E The requested operation could not be performed on an object named "<objektname>" that already exists.

Erläuterung: "<objektname>" ist in der Datenbank bzw. dem Betriebssystem bereits vorhanden. Das Objekt kann eine Datei, eine Tabelle, eine Spalte, ein Index, ein Auslöser oder ein anderes Objekt sein.

Benutzerantwort: Stellen Sie sicher, daß Sie das Objekt bei dem Versuch, darauf zuzugreifen, richtig angeben. Wenn nötig, löschen Sie das Objekt.

GSE0010E The requested operation could not be performed on an object named "<objektname>" that might not exist.

Erläuterung: "<objektname>" ist in der Datenbank oder im Betriebssystem nicht vorhanden. Das Objekt kann eine Datei, eine Tabelle, eine Spalte, ein Index, ein Auslöser oder ein anderes Objekt sein.

Benutzerantwort: Stellen Sie sicher, daß Sie eine ausreichende Berechtigung zum Zugreifen auf das Objekt haben. Wenn Sie diese Berechtigung haben und das Objekt nicht vorhanden ist, müssen Sie es erstellen.

GSE0011E Spatial Extender could not disable or unregister object "<objektname>".

Erläuterung: "<objektname>" hängt von einem anderen Objekt ab. "<objektname>" kann ein räumliches Bezugssystem, eine Schicht, ein Geocodierer oder ein anderes Objekt sein.

Benutzerantwort: Schlagen Sie in der Dokumentation nach, um festzustellen, von welcher Art von Objekt "<objektname>" abhängen kann. Entfernen Sie anschließend das spezifische Objekt, von dem "<objektname>" abhängt.

GSE0012E Spatial Extender could not process your request because the fully qualified spatial column "<schichtschema.schichtname.schichtspalte>" is not registered as a table layer.

Erläuterung: Die vollständig qualifizierte räumliche Spalte "<schichtschema.schichtname-.schichtspalte>" muß als Tabellenschicht registriert werden, bevor Sie bestimmte ihr zugeordnete Operationen ausführen können (z. B. das Aktivieren ihres Index, das Aktivieren eines Geocodierers zum Ausfüllen im Stapelbetrieb oder zum automatischen Aktualisieren).

Benutzerantwort: Stellen Sie sicher, daß die vollständig qualifizierte räumliche Spalte "<schichtschema.schichtname.schichtspalte>" als Tabellenspalte registriert ist. Überprüfen Sie hierzu die Sicht DB2GSE.GEOMETRY_COLUMNS im Spatial Extender-Katalog. Stellen Sie außerdem sicher, daß die Tabelle, die diese Spalte enthält, entsprechende gültige Attributspalten enthält.

GSE0013E The database is not enabled for spatial operations.

Erläuterung: Die Datenbank wurde nicht für räumliche Operationen aktiviert. Der Spatial Extender-Katalog ist deshalb nicht vorhanden.

Benutzerantwort: Aktivieren Sie die Datenbank für räumliche Operationen.

GSE0014E The database has already been enabled for spatial operations.

Erläuterung: Die Datenbank wurde bereits für räumliche Operationen aktiviert.

Benutzerantwort: Stellen Sie sicher, daß die Datenbank wie erwartet aktiviert wurde. Inaktivieren Sie die Datenbank gegebenenfalls.

GSE0498E The following error occurred: "<fehlernachricht>".

GSE0499W Spatial Extender issued the following warning: "<warnung>".

GSE0500E The operation mode that you specified ("<betriebsmodus>") is invalid.

Erläuterung: Der angegebene Modus wird von der angeforderten Operation nicht unterstützt.

Benutzerantwort: Schlagen Sie in der Dokumentation nach, um festzustellen, welche Modi von der Operation unterstützt werden.

GSE1001E Spatial Extender was unable to register a view layer that is named "<schemaname.sichtname.spaltenname>" and that is based on spatial column "<schemaname.tabellenname.spaltenname>".

Erläuterung: Die angegebene räumliche Spalte ("<schemaname.tabellename.spaltenname>") wurde nicht als Tabellenschicht registriert.

Benutzerantwort: Registrieren Sie die Spalte "<schemaname.tabellename.spaltenname>" als Tabellenschicht.

GSE1002E Spatial Extender was unable to register a view layer that is named "<schemaname.sichtname.spaltenname>" and that is based on table "<schemaname.tabellename>".

Erläuterung: Die Tabelle, die Sie in ("<schemaname.tabellenname>") angegeben haben, liegt der Sicht "<schemaname.sichtname.spaltenname>" weder direkt noch indirekt zugrunde.

Benutzerantwort: Stellen Sie fest, welches die Basistabelle für die Sicht "<schemaname.sichtname.spaltenname>" ist, und geben Sie diese Tabelle an.

GSE1003E Spatial Extender was unable to access a column named "<spaltenname>" in a table or view named "<schemaname.objektname>".

Erläuterung: Die Tabelle oder Sicht "<schemaname.objektname>" enthält keine Spalte mit dem Namen "<spaltenname>".

Benutzerantwort: Überprüfen Sie Definition der

Tabelle oder Sicht “<schemaname.objektname>”, um den richtigen Namen der gewünschten Spalte festzustellen.

GSE1004E Spatial Extender was unable to register the fully qualified spatial column “<schemaname.tabellenname.spaltenname>” as a table layer.

Erläuterung: Die Spalte “<schemaname.tabellenname.spaltenname>” hat keinen Typ räumlicher Daten oder ist nicht einer Basistabelle zugeordnet.

Benutzerantwort: Definieren Sie einen Typ räumlicher Daten für die Spalte “<schemaname-.tabellenname.spaltenname>”, oder stellen Sie sicher, daß diese Spalte Teil einer lokalen Basistabelle ist.

GSE1005E The spatial reference system (“<sichtschicht-räumliche-bezugs-ID>”) that you specified for a view layer differs from the spatial reference system (“<tabellenschicht-räumliche-bezugs-ID>”) that is used for this layer’s underlying table layer.

Erläuterung: Ein räumliches Bezugssystem einer Sichtschicht muß dasselbe sein wie das der Tabelle zugrundeliegende räumliche Bezugssystem.

Benutzerantwort: Geben Sie das der Tabellenschicht zugrundeliegende räumliche Bezugssystem für die Sichtschicht an.

GSE1006E Because “<räumliche-bezugs-ID>” is an invalid spatial reference system ID, Spatial Extender was unable to register the layer that you requested.

Erläuterung: Das angegebene räumliche Bezugssystem (“<räumliche-bezugs-ID>”) wurde nicht aktiviert oder registriert.

Benutzerantwort: Aktivieren oder registrieren Sie das räumliche Bezugssystem. Übergeben Sie anschließend Ihre Anforderung zur Registrierung der Schicht erneut.

GSE1007E An SQL error (SQLSTATE “<sqlstatus>”) might have occurred when Spatial Extender tried unsuccessfully to add a spatial column (“<spaltenname>”) to table “<schemaname.tabellenname>”.

Benutzerantwort: Schlagen Sie die Nachricht zu SQLSTATE “<sqlstatus>” nach.

GSE1008E DB2 Spatial Extender was unable to register a view layer“<schichtschema.schichtname.schichtspalte>” because the spatial data type “<schichtspaltentyp>” of the view layer does not match the spatial data type “<geo-spaltentyp>” of the underlying table layer “<geoschema.geo-name.geo-spalte>”.

Erläuterung: Der Typ von räumlichen Daten einer Sichtspalte “<schichtschema.schichtname-.schichtspalte>” muß dem Typ räumlicher Daten der Tabellenschicht “<geo-schema.geo-name.geospalte>” entsprechen, die der Schicht zugrundeliegt. Die Inkonsistenz zwischen diesen beiden Datentypen bewirkt eine Doppeldeutigkeit bei der Verarbeitung räumlicher Daten.

Benutzerantwort: Stellen Sie sicher, daß die Typen von räumlichen Daten der Sichtschicht und der ihr zugrundeliegende Tabellenschicht identisch sind.

GSE1020E “<’räumliche-bezugs-ID>” is an invalid spatial reference system ID.

Erläuterung: Ein räumliches Bezugssystem mit der Kennung “<’räumliche-bezugs-ID>” wurde nicht aktiviert.

Benutzerantwort: Stellen Sie sicher, daß der angegebene räumliche Bezug aktiviert wurde.

GSE1021E Spatial Extender could not enable spatial reference system "<räumliche-bezugs-ID>" because the corresponding spatial coordinate system ID "<räumliche-koordinaten-ID>" is invalid.

Erläuterung: Ein Koordinatensystem mit der Kennung "<räumliche-koordinaten-ID>" ist im Spatial Extender-Katalog nicht definiert.

Benutzerantwort: Überprüfen Sie die Kennung des Koordinatensystems "<räumliche-koordinaten-ID>". Überprüfen Sie hierzu die Sicht DB2GSE.COORD_REF_SYS im Spatial Extender-Katalog.

GSE1030E Because "<schemaname.tabellenname>" is not a base table, Spatial Extender could not enable a geocoder for it.

Erläuterung: Das Objekt, das die zu geocodierenden Quellendaten enthält, muß eine Basistabelle sein.

Benutzerantwort: Stellen Sie sicher, daß die Spalten mit den zu geocodierenden Daten Teil einer Basistabelle sind.

GSE1031E Spatial Extender could not enable geocoder "<geocodierer-ID>" to operate automatically in create mode for layer "<schichtschema.schichtname.schichtspalte>".

Erläuterung: Mögliche Erklärungen sind:

- Der Geocodierer ist bereits aktiviert für eine automatische Aktualisierung der Schicht "<schichtschema.schichtname.schichtspalte>".
- Der Geocodierer wurde für diese Schicht vorübergehend ungültig gemacht.
- Es wurden keine Spalten für Quellendaten für diese Schicht definiert.

Benutzerantwort: Wenn der Geocodierer vorübergehend ungültig gemacht wurde, aktivieren Sie ihn, so daß er automatisch im Modus ″Recreate″ arbeitet.

GSE1032E Spatial Extender could not enable geocoder "<geocodierer-ID>" to operate automatically in recreate mode for layer "<schichtschema.schichtname.schichtspalte>".

Erläuterung: Mögliche Erklärungen sind:

- Der Geocodierer ist bereits aktiviert für eine automatische Aktualisierung der Schicht "<schichtschema.schichtname.schichtspalte>".
- Der Geocodierer wurde für diese Schicht nicht zuvor ungültig gemacht.
- Es wurden keine Spalten für Quellendaten für diese Schicht definiert.

Benutzerantwort: Wenn der Geocodierer zuvor im Freigabemodus inaktiviert wurde oder überhaupt nicht für diese Schicht definiert wurde, aktivieren Sie ihn, so daß er automatisch im Modus ″Create″ arbeitet.

GSE1033E An SQL error occurred when Spatial Extender tried to add triggers to a table that contains the column for layer "<schichtschema.schichtname.schichtspalte>" (SQLSTATE "<sqlstatus>").

Erläuterung: Der Zweck der Auslöser ist das Aufrechterhalten der Datenintegrität zwischen den Attributspalten, aus denen die Eingabe des Geocodierers stammt, und der räumlichen Spalte, in die seine Ausgabe geht. Der SQL-Fehler trat auf, als DB2 versuchte, diese Auslöser zu erstellen.

Benutzerantwort: Schlagen Sie die Nachricht zu SQLSTATE "<sqlstatus>" nach.

GSE1034E Spatial Extender could not disable geocoder "<geocodierer-ID>" in drop mode for layer "<schichtschema.schichtname.schichtspalte>".

Erläuterung: Mögliche Erklärungen sind:

- Der Geocodierer wurde nicht aktiviert für eine automatische Aktualisierung der Schicht "<schichtschema.schichtname.schichtspalte>".

- Der Geocodierer wurde im Freigabemodus inaktiviert.

Benutzerantwort: Ermitteln Sie den Status des Geocodierers, bevor Sie versuchen, ihn zu inaktivieren. Wurde er registriert? War er aktiviert? Entscheiden Sie anschließend, ob er im Freigabemodus inaktiviert werden muß. Wenn er beispielsweise überhaupt nie aktiviert wurde, muß er auch nicht inaktiviert werden.

GSE1035E Spatial Extender could not disable geocoder "<geocodierer-ID>" in invalidate mode for layer "<schichtschema.schichtname. schichtspalte>".

Erläuterung: Mögliche Erklärungen sind:
- Der Geocodierer wurde nicht aktiviert für eine automatische Aktualisierung der Schicht "<schichtschema.schichtname.schichtspalte>".
- Der Geocodierer wurde im ungültigen Modus oder im Freigabemodus inaktiviert.

Benutzerantwort: Ermitteln Sie den Status des Geocodierers, bevor Sie versuchen, ihn zu inaktivieren. Wurde er registriert? War er aktiviert? Entscheiden Sie anschließend, ob er im ungültigen Modus inaktiviert werden muß. Wenn er beispielsweise bereits im ungültigen Modus inaktiviert wurde, muß er in diesem Modus nicht ein zweites Mal inaktiviert werden.

GSE1036E An SQL error occurred when Spatial Extender tried to drop triggers from a table that contains the column for layer "<schichtschema.schichtname-.schichtspalte>" (SQLSTATE "<sqlstatus>").

Erläuterung: Die Auslöser wurden erstellt, um die Datenintegrität zwischen den Attributspalten, aus denen die Eingabe des Geocodierers stammt, und der räumlichen Spalte, in die seine Ausgabe geht, aufrechtzuerhalten. Der SQL-Fehler trat auf, als DB2 versuchte, diese Auslöser freizugeben.

Benutzerantwort: Schlagen Sie die Nachricht zu SQLSTATE "<sqlstatus>" nach.

GSE1037E Spatial Extender could not geocode source data for table layer "<schichtschema.schichtname-.schichtspalte>", possibly because an incorrect value "<anzahl-der-attribute>" was assigned to the argument that specifies how many attribute columns are to provide source data for this layer.

Erläuterung: Die Anzahl der Attributspalten zu dieser Schicht wurde falsch angegeben, oder der Name einer oder mehrerer dieser Spalten wurde falsch angegeben.

Benutzerantwort: Stellen Sie sicher, daß diese Schicht mit der richtigen Anzahl und den richtigen Namen der zugeordneten Attributspalten registriert wurde, oder überprüfen Sie die Richtigkeit der Ein- und Ausgabedaten für den Geocodierer.

GSE1038E An SQL error occurred when Spatial Extender tried to geocode source data for table layer "<schichtschema.schichtname. schichtspalte>" in batch mode (SQLSTATE "<sqlstatus>").

Benutzerantwort:
- Schlagen Sie die Nachricht zu SQLSTATE "<sqlstatus>" nach.
- Stellen Sie sicher, daß der Inhalt und das Argument primaryUDF dieser Schicht richtig definiert wurden.

GSE1050E The grid size that you specified ("<gittergröße>") is invalid for the first grid level.

Erläuterung: Sie haben Null oder eine negative Zahl als Gittergröße für die erste Gitterstufe angegeben.

Benutzerantwort: Geben Sie eine positive Zahl für die Gittergröße an.

GSE1051E The grid size that you specified ("<gittergröße>") is invalid for the second and third grid levels.

Erläuterung: Sie haben eine negative Zahl als Gittergröße für die zweite Gitterstufe angegeben.

Benutzerantwort: Geben Sie Null oder eine positive Zahl für die Gittergröße an.

GSE1052E An SQL error occurred when the Spatial Extender tried to create spatial index "<indexschema-.indexspalte>" for a table layer "<schichtschema.schichtname-.schichtspalte>" (SQLSTATE "<sqlstatus>").

Benutzerantwort:

- Stellen Sie sicher, daß der räumliche Index richtig angegeben wurde und daß der räumlichen Spalte kein Index zugeordnet ist.
- Schlagen Sie die SQLSTATE zugeordnete Nachricht "<sqlstatus>" nach.

GSE1500I Source record "<datensatznummer>" was successfully geocoded.

Erläuterung: Ein Datensatz mit attributiven Daten wurde erfolgreich geocodiert.

GSE1501W Source record "<datensatznummer>" was not geocoded.

Erläuterung: Die Genauigkeitsstufe war zu hoch.

Benutzerantwort: Führen Sie eine Geocodierung mit einer niedrigeren Genauigkeitsstufe durch.

GSE1502W Source record "<datensatznummer>" was not found.

Benutzerantwort: Stellen Sie fest, ob der Datensatz in der Datenbank vorhanden ist.

GSE2001E The specified transfer file ("<dateiname>") is not valid.

Benutzerantwort: Stellen Sie sicher, daß die angegebene Datei eine SDE-Übertragungsdatei ist und der Pfadname richtig angegeben wurde.

GSE2002E The supplied SQL WHERE clause ("<SQL-where-klausel>") is not valid.

Benutzerantwort: Überprüfen Sie die WHERE-Klausel auf korrekte SQL-Syntax, Schreibfehler und ungültige Spaltennamen.

GSE2003E The supplied shape value is not legal.

Benutzerantwort: Stellen Sie sicher, daß die angegebene Form dem angegebenen Typ der räumlichen Spalte entspricht. Wenn die Typen übereinstimmen oder kompatibel sind, ist die Form der Geometrie nicht zulässig. Überprüfen Sie, ob überlappende Polygone, Einzelpunktbogen etc. vorliegen.

GSE2004E The transfer file schema is incompatible with the schema of the specified layer.

Benutzerantwort: Stellen Sie sicher, daß die Namen für das Schema und die Schicht richtig angegeben wurden. Stimmen die Schemata nicht überein, laden Sie die Daten als neue Tabelle, und lösen Sie die Schemadifferenzen auf.

GSE2005E The transfer file geometry type is incompatible with the geometry type of the specified layer.

Benutzerantwort: Stellen Sie sicher, daß die Namen für das Schema und die Schicht richtig angegeben wurden.

GSE2006E An I/O error for a file named "<dateiname>" has occurred.

Benutzerantwort: Stellen Sie sicher, daß die Datei vorhanden ist, daß Sie eine ausreichende Zugriffsberechtigung auf die Datei haben und daß die Datei nicht von einem anderen Benutzer verwendet wird.

GSE2007E An attribute conversion error has occurred.

Benutzerantwort: Stellen Sie sicher, daß alle Attributtypen in der Tabelle unterstützt werden - BLOB-Daten werden beispielsweise in Formdateien nicht unterstützt. Überprüfen Sie die Angabe außerdem auf Datenwerte außerhalb des zulässigen Bereichs oder auf ungültige Datenwerte wie beispielsweise falsche Datumsangaben.

GSE2008E The import/export function has run out of memory.

Benutzerantwort: Stellen Sie sicher, daß genügend Speicher zur Verfügung steht.

Kapitel 11. Katalogsichten

Die Katalogsichten des DB2 Spatial Extender enthalten Metadaten zu folgenden Informationen:

- Koordinatensysteme, die verwendet werden können. Informationen über Kennungen und Anmerkungstexten zu dieses Systemen finden Sie im Abschnitt „DB2GSE.COORD_REF_SYS".
- Räumliche Spalten, die als Schichten registriert wurden. Informationen zu den Namen dieser Spalten, den Datentypen und den zugeordneten räumlichen Bezugssystemen finden Sie im Abschnitt „DB2GSE.GEOMETRY_COLUMNS" auf Seite 116.
- Geocodierer, die Sie verwenden können. Informationen zu den Kennungen dieser Geocodierer und den Regionen, die die von diesen Geocodierern verarbeiteten Standorte enthalten, finden Sie im Abschnitt „DB2GSE.SPATIAL_GEOCODER" auf Seite 117.
- Räumliche Bezugssysteme, die Sie verwenden können. Informationen zu den Kennungen und ihren Beschreibungen finden Sie im Abschnitt „DB2GSE.SPATIAL_REF_SYS" auf Seite 118.

DB2GSE.COORD_REF_SYS

Wenn Sie eine Datenbank für räumliche Operationen aktivieren, registriert der DB2 Spatial Extender die zur Verfügung stehenden Koordinatensysteme in einer Katalogtabelle. Ausgewählte Spalten aus dieser Tabelle bilden die Katalogsicht DB2GSE.COORD_REF_SYS, die in Tabelle 34 beschrieben ist.

Tabelle 34. Spalten in der Katalogsicht DB2GSE.COORD_REF_SYS

Name	Datentyp	Null zulässig?	Inhalt
SCID	INTEGER	Nein	Eindeutige numerische Kennung für dieses Koordinatensystem.
SC_NAME	VARCHAR(64)	Nein	Name dieses Koordinatensystems
AUTH_NAME	VARCHAR(256)	Ja	Name der Organisation, die mit diesem Koordinatensystem arbeitet, z. B. die European Petroleum Survey Group (EPSG).
AUTH_SRID	INTEGER	Ja	Eine numerische Kennung, die diesem Koordinatensystem von der in der Spalte AUTH_NAME angegebenen Organisation zugeordnet wurde.
DESC	VARCHAR(256)	Ja	Beschreibung dieses Koordinatensystems.

Tabelle 34. Spalten in der Katalogsicht DB2GSE.COORD_REF_SYS (Forts.)

Name	Datentyp	Null zulässig?	Inhalt
SRTEXT	VARCHAR(2048)	Nein	Anmerkungstext für dieses Koordinatensystem.

DB2GSE.GEOMETRY_COLUMNS

Wenn Sie eine Schicht erstellen, registriert der DB2 Spatial Extender diese durch Aufzeichnen ihrer Kennung und ihrer Informationen in einer Katalogtabelle. Ausgewählte Spalten aus dieser Tabelle bilden die Katalogsicht DB2GSE.GEOMETRY_COLUMNS, die in Tabelle 35 beschrieben ist.

Tabelle 35. Spalten in der Katalogsicht DB2GSE.GEOMETRY_COLUMNS

Name	Datentyp	Null zulässig?	Inhalt
LAYER_CATALOG	VARCHAR(30)	Ja	Vollständig qualifizierter Name dieser Schicht.
LAYER_SCHEMA	VARCHAR(30)	Nein	Schema der Tabelle oder Sicht, die die als diese Schicht registrierte Spalte enthält.
LAYER_NAME	VARCHAR(128)	Nein	Name der Tabelle oder Sicht, die die als diese Schicht registrierte Spalte enthält.
LAYER_COLUMN	VARCHAR(30)	Nein	Name der Spalte, die als diese Schicht registriert wurde.
GEOMETRY_TYPE	INTEGER	Nein	Datentyp der Spalte, die als diese Schicht registriert wurde.
SRID	INTEGER	Nein	Kennung des räumlichen Bezugssystems, das für die Werte in der Spalte, die als diese Schicht registriert wurde, verwendet wird.
STORAGE_TYPE	INTEGER	Ja	Informationen dazu, wie DB2 die Werte in der Spalte speichert, die als diese Schicht registriert wurde. Daten in STORAGE_TYPE können z. B. darauf hinweisen, daß die Werte als große Objekte (LOBs) oder als Exemplare abstrakter Datentypen (ADTs) gespeichert wurden.

DB2GSE.SPATIAL_GEOCODER

Die verfügbaren Geocodierer sind in einer Katalogtabelle registriert. Ausgewählte Spalten aus dieser Tabelle bilden die Katalogsicht DB2GSE.SPATIAL_GEOCODER, die in Tabelle 36 beschrieben ist.

Tabelle 36. Spalten in der Katalogsicht DB2GSE.SPATIAL_GEOCODER

Name	Datentyp	Null zulässig?	Inhalt
GCID	INTEGER	Nein	Numerische Kennung dieses Geocodierers.
GC_NAME	VARCHAR(64)	Nein	Kurzbeschreibung dieses Geocodierers.
VENDOR_NAME	VARCHAR(128)	Nein	Name des Herstellers, der diesen Geocodierer zur Verfügung gestellt hat.
PRIMARY_UDF	VARCHAR(256)	Nein	Vollständig qualifizierter Name dieses Geocodierers.
PRECISION_LEVEL	INTEGER	Nein	Die Genauigkeit, mit der die Quellendaten den entsprechenden Bezugsdaten entsprechen müssen, damit sie vom Geocodierer erfolgreich verarbeitet werden können.
VENDOR_SPECIFIC	VARCHAR(256)	Ja	Pfad und Name einer Datei, die ein Hersteller verwenden kann, um spezielle Parameter festzulegen, die von diesem Geocodierer unterstützt werden.
GEO_AREA	VARCHAR(256)	Ja	Geographischer Bereich, der die zu geocodierenden Standorte enthält.
DESCRIPTION	VARCHAR(256)	Ja	Anmerkungen des Herstellers.

DB2GSE.SPATIAL_REF_SYS

Wenn Sie ein räumliches Bezugssystem erstellen, registriert der DB2 Spatial Extender dieses durch Aufzeichnen seiner Kennung und seiner Informationen in einer Katalogtabelle. Ausgewählte Spalten aus dieser Tabelle bilden die Katalogsicht DB2GSE.SPATIAL_REF_SYS, die in Tabelle 37 beschrieben ist.

Tabelle 37. Spalten in der Katalogsicht DB2GSE.SPATIAL_REF_SYS

Name	Datentyp	Null zulässig?	Inhalt
SRID	INTEGER	Nein	Benutzerdefinierte Kennung für dieses räumliche Bezugssystem.
SR_NAME	VARCHAR(64)	Nein	Name dieses räumlichen Bezugssystems.
SCID	INTEGER	Nein	Numerische Kennung für das Koordinatensystem, das diesem räumlichen Bezugssystem zugrundeliegt.
SC_NAME	VARCHAR(64)	Nein	Name des Koordinatensystems, das diesem räumlichen Bezugssystem zugrundeliegt.
AUTH_NAME	VARCHAR(256)	Ja	Name der Organisation, die die Standards für dieses räumliche Bezugssystem festlegt.
AUTH_SRID	INTEGER	Ja	Die Kennung, die die in der Spalte AUTH_NAME angegebene Organisation diesem räumlichen Bezugssystem zuordnet.
SRTEXT	VARCHAR(2048)	Nein	Anmerkungstext für dieses räumliche Bezugssystem.

Kapitel 12. Räumliche Indizes

Da räumliche Spalten zweidimensionale geographische Daten enthalten, erfordern Anwendungen, die diese Spalten abfragen, eine Indexstrategie, die alle Geometrien innerhalb eines bestimmten Bereichs schnell identifiziert. Aus diesem Grund bietet der DB2 Spatial Extender den dreistufigen räumlichen Index auf der Basis eines Gitters.

Dieses Kapitel beschreibt diese Art von Index und definiert Richtlinien zur Verwendung dieses Index. Folgende Themen werden hierbei behandelt:

Fragment eines Beispielprogramms

Das folgende Beispiel zeigt, wie ein Index in SQL erstellt und verwendet wird. Sie können weitere Informationen zu den Befehlen CREATE INDEX und CREATE INDEX EXTENSION im Handbuch *SQL Reference* nachschlagen. Beachten Sie, daß Sie nach dem Erstellen des Index Standard-DDL- und DML-Anweisungen verwenden können, die mit den räumlichen Funktionen und Prädikaten arbeiten.

```
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)
```

B-Baumstruktur-Indizes

Die Technologie der räumlichen Indexierung basiert auf dem herkömmlichen hierarchischen B-Baumstruktur-Index, weist jedoch erhebliche Unterschiede auf. Der räumliche Index verwendet eine *Gitterindexierung* zum Indexieren zweidimensionaler räumlicher Spalten. Der B-Baumstruktur-Index kann nur eindimensionale Daten verarbeiten und nicht mit GIS-Informationen verwendet werden. Dieser Abschnitt beschreibt, wie ein B-Baumstruktur-Index strukturiert und verwendet wird.

Die oberste Ebene eines B-Baumstruktur-Index, die als Root-Knoten bezeichnet wird, enthält einen Schlüssel für jeden Knoten auf der nächsten Ebene. Der Wert jedes Schlüssels ist der größte vorhandene Schlüsselwert für den entsprechenden Knoten auf der nächsten Ebene. Je nach der Anzahl der Werte in der Basistabelle sind eventuell mehrere Zwischenknoten erforderlich. Diese Knoten bilden eine Brücke zwischen dem Root-Knoten und den "Blattknoten", die die eigentlichen Zeilen-IDs der Basistabelle enthalten.

Der Datenbank-Manager durchsucht einen B-Baumstruktur-Index ab dem Root-Knoten. Anschließend fährt er mit den Zwischenknoten fort, bis er den Blattknoten mit der Zeilen-ID der Basistabelle erreicht.

Der B-Baumstruktur-Index kann nicht auf eine räumliche Spalte angewandt werden, da die Zweidimensionalität der räumlichen Spalte die Struktur eines räumlichen Index erfordert. Aus dem gleichen Grund können Sie keinen räumlichen Index auf eine nicht räumliche Spalte anwenden. Darüber hinaus kann ein räumlicher Index nicht auf eine zusammengesetzte Spalte irgendeiner Art angewandt werden.

Möglichkeiten zum Erstellen eines räumlichen Index

Ein räumlicher Index kann auf verschiedene Arten erstellt werden:

- Durch Definieren im Fenster **Create Spatial Index**. Anleitungen hierzu finden Sie in „Kapitel 6. Räumliche Indizes erstellen" auf Seite 55.
- Durch Aufrufen der gespeicherten Prozedur db2gse.gse_enable_idx in einem Anwendungsprogramm. Informationen zu dieser gespeicherten Prozedur finden Sie in „Kapitel 9. Gespeicherte Prozeduren" auf Seite 71.
- Durch Eingeben des Befehls **db2 create index** mit der Funktion **spatial_index** in der USING-Klausel. Beispiel:

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

Zur Verwendung räumlicher Daten muß der Datenbank-Designer ihre relative Größenverteilung verstehen. Der Designer muß die optimale Größe und die Anzahl der Gitterstufen festlegen, mit denen der räumliche Index erstellt werden soll.

Die Gitterstufen <Gitterstufe 1>, <Gitterstufe 2> und <Gitterstufe 3> werden durch Vergrößern der Zellengröße eingegeben. Somit muß die zweite Schicht eine höhere Zellengröße haben als die erste und die dritte Schicht eine höhere als die zweite. Die erste Gitterstufe ist verbindlich; Sie können die zweite und dritte mit einem Nullwert doppelter Genauigkeit (0.0e0) inaktivieren.

Wie ein räumlicher Index erstellt wird

Ein räumlicher Index wird durch Verwendung von *Umschlägen* erstellt. Der Umschlag ist selbst eine Geometrie und stellt den Minimal- und Maximalbereich für X und Y einer Geometrie dar. Für die meisten Geometrien ist der Umschlag ein Kasten; für horizontale und vertikale Linienfolgen ist es jedoch eine Zwei-Punkt-Linienfolge. Für Punkte ist der Umschlag selbst ein Punkt. Weitere Informationen zu Umschlägen finden Sie im Abschnitt „Umschlag" auf Seite 135.

Der räumliche Index wird in einer räumlichen Spalte konstruiert; hierzu werden ein oder mehrere Einträge für die Schnittpunkte der Umschläge der einzelnen Geometrien mit dem Gitter vorgenommen. Ein Schnittpunkt wird als interne ID der Geometrie und als Mindestkoordinaten für X und Y der geschnittenen Gitterzellen aufgezeichnet. Das Polygon in Abb. 7 auf Seite 122 schneidet beispielsweise das Gitter an den Koordinaten (20,30), (30,30), (40,30), (20,40), (30,40), (40,40), (20,50), (30,50) und (40,50). Siehe Tabelle 38 auf Seite 123 für Mindestwerte der X- und Y-Koordinaten für alle Geometrien in Abb. 7 auf Seite 122.

Wenn mehrere Gitterstufen vorhanden sind, versucht der DB2 Spatial Extender, die niedrigste mögliche Gitterstufe zu verwenden. Wenn eine Geometrie

vier oder mehr Gitterzellen auf einer angegebenen Ebene geschnitten hat, wird sie auf die nächsthöhere Stufe hochgestuft. Bei einem räumlichen Index mit den drei Gitterstufen 10.0e0, 100.0e0 und 1000.0e0 schneidet der DB2 Spatial Extender daher zunächst jede Geometrie mit dem Gitter der Ebene 10.0e0. Wenn eine Geometrie vier oder mehr 10.0e0-Gitterzellen schneidet, wird sie hochgestuft und mit dem Gitter der Ebene 100.0e0 geschnitten. Wenn sich vier oder mehr Schnittpunkte auf der Ebene 100.0e0 ergeben, wird die Geometrie auf die Ebene 1000.0e0 hochgestuft. Auf der Ebene 1000.0e0 müssen die Schnittpunkte im räumlichen Index eingegeben werden, da dies die höchste mögliche Ebene ist.

Abb. 7 verdeutlicht, wie vier verschiedene Arten von Geometrien ein 10.0e-Gitter schneiden. Alle 23 Schnittpunkte für die vier Geometrien werden im räumlichen Index aufgezeichnet.

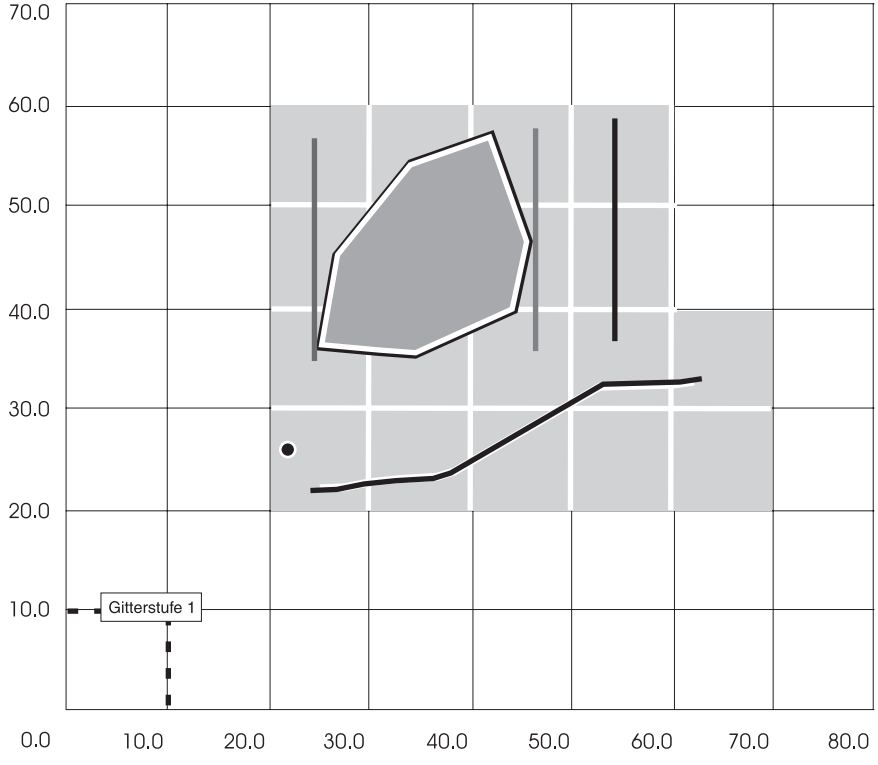


Abbildung 7. Anwendung einer 10.0e0-Gitterstufe

Tabelle 38 listet die Geometrien und ihre entsprechenden Gitterschnittpunkte auf. Die Umschläge der vier verschiedenen Geometrietypen schneiden das 10.0e-Gitter. Die Mindestkoordinaten für X und Y jeder geschnittenen Gitterzelle werden in dem räumlichen Index eingegeben.

Tabelle 38. Die 10.0e0-Gitterzelleneinträge für die Beispielgeometrien

Geometrie	Gitter X	Gitter Y
Polygon	20.0	30.0
Polygon	30.0	30.0
Polygon	40.0	30.0
Polygon	20.0	40.0
Polygon	30.0	40.0
Polygon	40.0	40.0
Polygon	20.0	50.0
Polygon	30.0	50.0
Polygon	40.0	50.0
Vertikale Linienfolge	50.0	30.0
Vertikale Linienfolge	50.0	40.0
Vertikale Linienfolge	50.0	50.0
Punkt	20.0	20.0
Horizontale Linienfolge	20.0	20.0
Horizontale Linienfolge	30.0	20.0
Horizontale Linienfolge	40.0	20.0
Horizontale Linienfolge	50.0	20.0
Horizontale Linienfolge	60.0	20.0
Horizontale Linienfolge	20.0	30.0
Horizontale Linienfolge	30.0	30.0
Horizontale Linienfolge	40.0	30.0
Horizontale Linienfolge	50.0	30.0
Horizontale Linienfolge	60.0	30.0

Abb. 8 zeigt, wie die Anzahl der Schnittpunkte erheblich auf acht reduziert werden kann durch Hinzufügen der Gitterstufen 30.0e0 und 60.0e0. In diesem Fall werden das als Geometrie 1 gekennzeichnete Polygon auf die Gitterstufe 30.0e0 und die als Geometrie 4 gekennzeichnete Linienfolge auf die Gitterstufe 60.0e0 hochgestuft. Statt der neun und zehn Schnittpunkte, die die Geometrien auf der Stufe 10.0e0 hatten, haben sie nach dem Hochstufen nur noch zwei Schnittpunkte.

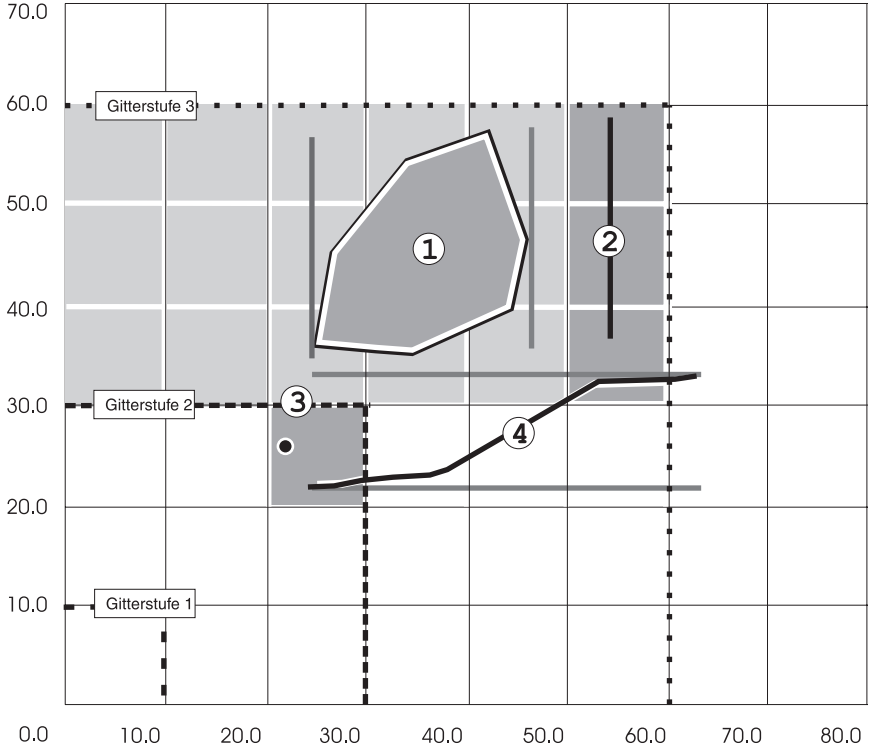


Abbildung 8. Auswirkung beim Hinzufügen der Gitterstufen 30.0e0 und 60.0e0. Der Umschlag des als Geometrie 1 gekennzeichneten Umschlags schneidet neun Gitterzellen. Der Umschlag der als Geometrie 2 vertikalen Linienfolge schneidet drei Gitterzellen. Der Umschlag des als Geometrie 3 gekennzeichneten Punkts schneidet nur eine Gitterzelle. Der Umschlag der als Geometrie 4 gekennzeichneten Linienfolge schneidet zehn Gitterzellen.

Der DB2 Spatial Extender verwendet die in der Anweisung CREATE INDEX angegebenen Gitterstufenparameter und prüft jedes räumliche Objekt, um die Koordinaten und die Anzahl der Gitterblöcke, in denen die Objekte enthalten sind, festzustellen. In Abb. 8 sind die Gitterstufen 10.0e0, 30.0e0 und 60.0e0 dargestellt mit zunehmenden Linienstärken und unterschiedlichen Grauschattierungen. Die Zellenschnittpunkte der vertikalen Linienfolgen- und des Punktumschlags werden in dem Index auf der Stufe 10.0e0 eingegeben, da beide weniger als vier Schnittpunkte erzeugen. Das Polygon schneidet neun 10.0e0-Gitterzellen und wird daher hochgestuft auf die 30.0e0-Stufe. Auf

dieser Stufe schneidet das Polygon zwei Gitterzellen, die in den Index eingegeben werden. Die als Geometrie 4 gekennzeichnete Linienfolge schneidet zehn 10.0e0-Gitterzellen und wird daher hochgestuft auf die 30.0e0-Gitterstufe. Auch auf dieser Stufe schneidet sie noch sechs Gitterzellen und wird daher erneut hochgestuft auf die 60.0e0-Gitterstufe, wo sie noch zwei Schnittpunkte generiert. Die 60.0e0-Gitterschnittpunkte der Linienfolge werden anschließend in den Index eingegeben. Hätte die Linienfolge auf dieser Ebene vier oder mehr Schnittpunkte generiert, wären diese dennoch in den Index eingetragen worden, da dies die höchste Stufe ist, auf die eine Geometrie hochgestuft werden kann.

Tabelle 39. Die Schnittpunkte der Geometrien im dreistufigen Index

Geometrie	Gitter X	Gitter Y
Die Schnittpunkte zwischen der vertikalen Linienfolge und dem Punkt auf der Stufe 1 (10.0e0-Gittergröße)		
2	50.0	30.0
2	50.0	40.0
2	50.0	50.0
3	20.0	20.0
Die Schnittpunkte des Polygons auf der Stufe 2 (30.0e0-Gittergröße)		
1	0.0	30.0
1	30.0	30.0
Die Schnittpunkte der Linienfolge auf der Stufe 3 (60.0e0-Gittergröße)		
4	0.0	0.0
4	60.0	0.0

Der DB2 Spatial Extender erstellt nicht wirklich eine Polygongitterstruktur. Der DB2 Spatial Extender manifestiert jede Gitterstufe parametrisch durch die Definition des Ursprungs am Abstand X,Y des räumlichen Bezugssystems der Spalten. Anschließend wird das Gitter in den positiven Koordinatenbereich erweitert. Mit einem parametrischen Gitter generiert der DB2 Spatial Extender die Schnittpunkte mathematisch.

Richtlinien zur Verwendung eines räumlichen Index

Der DB2 Spatial Extender verbessert mit Hilfe eines räumlichen Index die Leistung einer räumlichen Abfrage. Betrachten Sie die einfachste und wahrscheinlich häufigste räumliche Abfrage, die Feldabfrage. Diese Abfrage fordert den DB2 Spatial Extender auf, alle Geometrien zurückzugeben, die vollständig oder teilweise innerhalb eines benutzerdefinierten Felds liegen. Wenn kein Index vorhanden ist, muß der DB2 Spatial Extender alle Geometrien mit dem Feld vergleichen. Mit einem Index kann der DB2 Spatial Extender jedoch alle Indexeinträge finden, deren linke untere Koordinate größer oder gleich der des Feldes ist und deren obere rechte Koordinate kleiner oder gleich der des Feldes ist. Da der Index nach diesem Koordinatensystem geordnet ist, kann der DB2 Spatial Extender schnell eine Liste der Kandidatengeometrien abrufen. Dieser Prozeß wird als *erster Arbeitsgang* bezeichnet.

Ein *zweiter Arbeitsgang* ermittelt, ob die Umschläge der Kandidaten einen Kasten schneiden. Eine für den ersten Arbeitsgang qualifizierte Geometrie, deren Gitterzellenumschlag den Kasten schneidet, kann selbst einen Umschlag haben, der den Kasten nicht schneidet.

Ein *dritter Arbeitsgang* vergleicht die tatsächlichen Koordinaten des Kandidaten mit dem Kasten, um festzustellen, ob ein Teil der Geometrie tatsächlich in dem Kasten liegt. Dieser letzte und relativ komplexe Vergleichsprozeß arbeitet mit einer Liste von Kandidaten, die aus einer Untermenge der Gesamtpopulation bestehen, die durch die beiden ersten Arbeitsgänge erheblich reduziert wurde.

Alle räumlichen Abfragen mit Ausnahme der Funktion EnvelopesIntersect führen die drei Arbeitsgänge aus. Diese Funktion führt nur die beiden ersten Arbeitsgänge aus. Die Funktion EnvelopesIntersect wurde für Anzeigeoperationen konzipiert, die häufig ihre eigenen integrierten Ausschnittroutinen verwenden und die Unterteilung des dritten Arbeitsgangs nicht erfordern.

Gitterzellengröße auswählen

Die unregelmäßige Form der Geometrieumschläge kompliziert die Auswahl der Gitterzellengröße. Wegen dieser Unregelmäßigkeit schneiden manche Geometrieumschläge mehrere Gitter, während andere in eine einzige Gitterzelle passen. Andererseits schneiden manche Gitterzellen, je nach der räumlichen Verteilung der Daten, viele Geometrieumschläge.

Damit ein räumlicher Index gut funktioniert, ist die Auswahl der richtigen Anzahl und Größe der Gitter sehr wichtig. Betrachten Sie eine räumliche Spalte mit einer Geometrie einheitlicher Größe. In diesem Fall genügt eine einzige Gitterstufe. Beginnen Sie mit einer Gitterzellengröße, die den durchschnittlichen Geometrieumschlag umfaßt. Beim Testen Ihrer Anwendung stellen Sie eventuell fest, daß durch die Vergrößerung der Gitterzellengröße die Leistung Ihrer Abfragen verbessert wird. Dies liegt daran, daß jede Gitterzelle mehrere Geometrien enthält und der erste Arbeitsgang nicht qualifizierte Geometrien schneller verwerfen kann. Sie werden jedoch ebenfalls feststellen, daß die Leistung bei einer weiteren Vergrößerung der Zellengröße wieder nachläßt. Das liegt daran, daß der zweite Arbeitsgang schließlich mit mehr Kandidaten arbeiten muß.

Anzahl der Stufen auswählen

Wenn die Objekte, die Sie indexieren wollen, ungefähr gleich groß sind, können Sie mit einer einzigen Gitterstufe arbeiten. Dies ist zwar richtig; es enthalten jedoch nicht alle Spalten eine Geometrie der gleichen relativen Größe. Normalerweise können Geometrien räumlicher Spalten in Intervallen verschiedener Größen gruppiert werden. Denken Sie beispielsweise an ein Straßennetz, in dem die Geometrien in Straßen, Hauptstraßen und Fernstraßen unterteilt sind. Die Straßen sind alle ungefähr gleich lang und können in einer Intervallgröße gruppiert werden. Dies gilt auch für die Hauptstraßen und Fernstraßen. Die Straßen, die für ein Größenintervall stehen, können daher in der erste Gitterstufe gruppiert werden, die Hauptstraßen in der zweiten und die Fernstraßen in der dritten. Ein weiteres Beispiel bildet eine Spalte "Landkreisflächen", die Gruppen kleiner städtischer Flächen enthält, die von größeren ländlichen Flächen umgeben sind. In diesem Fall gibt es zwei Größenintervalle und zwei Gitterstufen, eine für die kleinen städtischen Flächen und eine andere für die größeren ländlichen Flächen. Diese Situationen sind sehr häufig und erfordern die Verwendung eines mehrstufigen Gitters.

Zur Auswahl der Zellengröße jeder Gitterstufe wählen Sie Gitterzellengrößen aus, die etwas größer sind als jedes Größenintervall. Testen Sie den Index durch Ausführen von Abfragen, gegen die räumlichen Spalten.

Jede zusätzliche Stufe erfordert eine zusätzliche Indexsuche. Versuchen Sie, die Gittergrößen leicht nach oben oder unten anzupassen, um festzustellen, ob dadurch eine Verbesserung der Leistung erzielt wird.

Kapitel 13. Geometrien und zugeordnete räumliche Funktionen

In diesem Kapitel werden die Informationseinheiten, die sogenannten *Geometrien* beschrieben, die aus Koordinaten bestehen und geographische Funktionen symbolisieren. Das Kapitel stellt außerdem räumliche Funktionen vor, die Geometrien als Eingabe verwenden und Ergebnisse zurückgeben, mit denen Sie geographische Eigenschaften einfacher analysieren und räumliche Daten zwischen geographischen Informationssystemen verschieben können. Folgende Themen werden hierbei behandelt:

- Das Wesen der Geometrien
- Merkmale der Geometrien; Funktionen, die Informationen zu diesen Merkmalen zurückgeben
- Exemplarfähige Geometrien; Funktionen, die darauf ausgeführt werden
- Funktionen zur Ausführung folgender Aufgaben:
  - Aufzeigen von Beziehungen und Vergleichen zwischen geographischen Merkmalen
  - Generieren von Geometrien
  - Umwandeln von Geometriewerten in importierbare und exportierbare Formate

Informationen zu Geometrien

Das Oxford American Dictionary definiert *Geometrie* als den „Zweig der Mathematik, der sich mit den Eigenschaften von und den Beziehungen zwischen Geraden, Winkeln, Flächen und Körpern beschäftigt." Am 11. August 1997 hat das Open GIS Consortium Inc. (OGC) in seiner Veröffentlichung *Open GIS Features for ODBC (SQL) Implementation Specification* eine andere Definition für den Begriff geprägt. Das Wort *Geometrie* wurde ausgewählt, um die geometrischen Eigenschaften zu kennzeichnen, mit deren Hilfe Kartographen seit mehr als einem Jahrtausend die Erde darstellen. Eine sehr abstrakte Definition dieser neuen Bedeutung der Geometrie wäre „ein Punkt oder eine Gruppierung von Punkten, die ein Merkmal der Oberfläche symbolisieren."

Im DB2 Spatial Extender wäre eine *operative* Definition der Geometrie „ein Modell einer geographischen Eigenschaft." Das Modell kann als Koordinaten der Eigenschaft ausgedrückt werden oder in manchen Fällen als visuelles Symbol. Das Modell umfaßt Informationen; die Koordinaten kennzeichnen beispielsweise die Position der Eigenschaft in bezug auf feste Referenzpunkte, und das Symbol umreißt ihre Form.

Darüber hinaus kann das Modell verwendet werden, um Informationen zu erstellen; die Funktion ST_Overlaps kann beispielsweise die Koordinaten von zwei benachbarten Regionen als Eingabe verwenden und als Ausgabe Informationen dazu zurückgeben, ob sich die Regionen überlappen oder nicht.

Die Koordinaten eines Merkmals, das von einer Geometrie symbolisiert wird, werden als *Merkmale* der Geometrie betrachtet. Verschiedene Arten von Geometrien haben ebenfalls Merkmale, z. B.:

- Ein *Innenbereich* steht für den Inhalt eines Merkmals, das von der Geometrie symbolisiert wird.
- Ein *Außenbereich* steht für den Bereich um das Merkmal herum.
- Eine *Begrenzung* steht für die Linie, an der der Inhalt aufhört und der Umgebungsbereich beginnt.

Diese und weitere Merkmale werden im Abschnitt „Merkmale von Geometrien und ihre zugeordneten Funktionen" auf Seite 132 beschrieben.

Die vom DB2 Spatial Extender unterstützten Geometrien bilden eine Hierarchie, wie in Abb. 9 auf Seite 131 dargestellt. Sechs Teile der Hierarchie sind exemplarfähig; sie können als visuelle Symbole ausgedrückt werden, die ebenfalls in der Abbildung dargestellt sind.

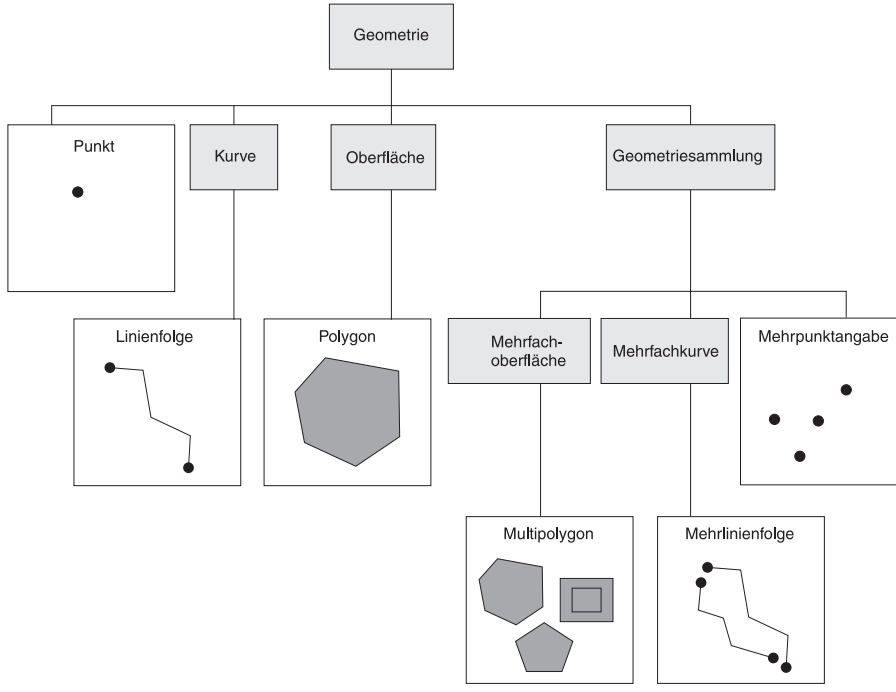


Abbildung 9. Hierarchie der vom DB2 Spatial Extender unterstützten Geometrien. Exemplarfähige Geometrien können als visuelle Symbole ausgedrückt werden. Diese Symbole werden unter den Namen dieser Geometrien angezeigt.

Wie in Abb. 9 angedeutet, ist eine Superklasse *geometry* der Root-Knoten der Hierarchie. Die Unterklassen sind in zwei Kategorien unterteilt: die Unterklassen der Basisgeometrien und die Unterklassen der Gruppen. Die Basisgeometrien umfassen:

- *Punkte* für die eindeutigen Merkmale, die den Ort belegen, an dem eine Ost-West-Koordinatenlinie (z. B. ein Breitengrad) eine Nord-Süd-Koordinate (z. B. einen Längengrad) schneidet. Angenommen, die Notation einer Karte mit großem Maßstab zeigt jede Stadt auf der Karte an einem Schnittpunkt eines Breiten- und Längengrads. Bei diesem Maßstab könnte jede Stadt durch einen Punkt symbolisiert werden.
- *Linienfolgen* für lineare geographische Merkmale (z. B. Straßen, Kanäle oder Rohrleitungen).
- *Polygone* für mehrseitige geographische Merkmale (z. B. Industriegebiete, Wälder oder Naturschutzgebiete).

Diese homogenen Gruppen umfassen:

- *Mehrpunktangaben* für mehrteilige Merkmale, deren Komponenten sich jeweils am Schnittpunkt einer Ost-West- und einer Nord-Süd-Koordinate befinden (z. B. eine Inselkette, deren einzelne Inseln sich an den Schnittpunkten eines Breiten- und Längengrades befinden).
- *Mehrlinienfolgen* für mehrteilige Merkmale aus linearen Einheiten der Komponenten (z. B. Fluß- oder Straßennetze).
- *Multipolygone* für mehrteilige Merkmale aus mehrseitigen Einheiten oder Komponenten (z. B. das Ackerland in einer bestimmten Region oder eine Seenplatte).

Homogene Gruppen sind, wie der Name schon andeutet, Gruppen von Basisgeometrien. Neben den gemeinsamen Basisgeometrieeigenschaften haben homogene Gruppen auch eigene Eigenschaften.

Die vom DB2 Spatial Extender unterstützten Typen von räumlichen Daten sind Implementierungen der in Abb. 9 auf Seite 131 dargestellten Geometrien. Eine Beschreibung dieser Datentypen finden Sie im Abschnitt „Informationen zu Typen von räumlichen Daten" auf Seite 36.

Merkmale von Geometrien und ihre zugeordneten Funktionen

In diesem Abschnitt werden die Merkmale von Geometrien beschrieben sowie die räumlichen Funktionen, die diesen Merkmalen zugeordnet sind. Der Abschnitt beginnt mit den Kernmerkmalen:

- Zu welcher Klasse eine Geometrie gehört
- X- und Y-Koordinaten

Außerdem erläutert der Abschnitt folgende Begriffe:

- Z-Koordinaten
- Maße
- Innenbereich, Begrenzung und Außenbereich einer Geometrie
- Die Qualität der Komplexität (einfach oder nicht einfach)
- Die Qualität des Inhalts (leer oder nicht leer)
- Umschlag einer Geometrie
- Dimension
- Die Kennung eines der Geometrie zugeordneten räumlichen Bezugssystems

Klasse

Jede Geometrie gehört zu einer Klasse in der in Abb. 9 auf Seite 131 dargestellten Hierarchie. Wie im Abschnitt „Informationen zu Geometrien" auf Seite 129 angegeben, gibt es sechs exemplarfähige Unterklassen in der Hierarchie — Punkte, Linienfolgen, Polygone, Mehrpunktangaben, Mehrlinienfolgen und Multipolygone. Die Superklasse und andere Unterklassen sind nicht exemplarfähig.

Die Funktion ST_GeometryType verwendet eine Geometrie und gibt eine exemplarfähige Unterklasse in Form einer Zeichenfolge zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_GeometryType" auf Seite 227. Die Funktion ST_IsValid verwendet eine Geometrie, die einem Datentyp ST_Geometry zugeordnet wurde. Die Funktion gibt 1 (TRUE) zurück, wenn die Geometrie gültig ist, und 0 (FALSE), wenn sie nicht gültig ist. Weitere Informationen hierzu finden Sie im Abschnitt „ST_IsValid" auf Seite 244.

X- und Y-Koordinaten

Ein *X-Koordinatenwert* kennzeichnet eine Position relativ zu einem Bezugspunkt im Osten oder Westen. Ein *X-Koordinatenwert* kennzeichnet eine Position relativ zu einem Bezugspunkt im Norden oder Süden. Weitere Informationen hierzu finden Sie in den Abschnitten „Das Wesen der räumlichen Daten" auf Seite 6 und „Informationen zu Koordinaten und räumlichen Bezugssystemen" auf Seite 27.

Z-Koordinaten

Manche Geometrien haben eine zugeordnete Höhe oder Tiefe. Jeder Punkt der Geometrie eines Merkmals kann eine wahlfreie Z-Koordinate enthalten, die eine Höhe oder Tiefe gegenüber der Erdoberfläche angibt.

Die Prädikatfunktion Is3d verwendet eine Geometrie und gibt 1 (TRUE) zurück, wenn die Funktion Z-Koordinaten hat und andernfalls 0 (FALSE). Weitere Informationen hierzu finden Sie im Abschnitt „Is3d" auf Seite 175.

Maße

Ein Maß ist ein Wert, der Informationen über ein geographisches Merkmal enthält, die zusammen mit den Koordinaten für die Position des Merkmals gespeichert werden. Angenommen, Sie stellen in Ihrem GIS Transportsysteme dar. Wenn Ihre Anwendung Werte verarbeiten soll, die lineare Entfernungen (Luftlinie) angeben, können Sie diese Werte zusammen mit den Koordinaten für die Position der Systeme speichern. Maße werden als Zahlen mit doppelter Genauigkeit gespeichert.

Das Prädikat IsMeasured verwendet eine Geometrie und gibt 1 (TRUE) zurück, wenn Sie Maße enthält und andernfalls 0 (FALSE). Weitere Informationen hierzu finden Sie im Abschnitt „IsMeasured" auf Seite 176.

Innenbereich, Begrenzung und Außenbereich

Alle Geometrien belegen eine Position im Raum; diese Position ist durch ihren Innenbereich, ihre Begrenzung und ihren Außenbereich gekennzeichnet. Der Außenbereich einer Geometrie ist der gesamte Raum, der nicht von der Geometrie belegt wird. Die Begrenzung einer Geometrie dient als Schnittstelle zwischen ihrem Innen- und ihrem Außenbereich. Der Innenbereich ist der von der Geometrie belegte Raum. Unterklassen übernehmen die Merkmale von Innenbereich und Außenbereich direkt; die Merkmale der Begrenzung unterscheiden sich jedoch für die einzelnen Unterklassen.

Die Funktion ST_Boundary verwendet eine Geometrie und gibt eine Geometrie zurück, die die Begrenzung der Quellengeometrie darstellt. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Boundary" auf Seite 197.

Einfach oder nicht einfach

Manche Unterklassen einer Geometrie (Linienfolge, Mehrpunktangaben und Mehrlinienfolgen) sind entweder einfach oder nicht einfach. Eine Unterklasse ist einfach, wenn sie alle topologischen Regeln der Unterklasse einhält; werden nicht alle diese Regeln eingehalten, so ist die Klasse nicht einfach. Eine Linienfolge ist einfach, wenn sie ihren eigenen Innenbereich nicht schneidet. Eine Mehrpunktangabe ist einfach, wenn keines ihrer Elemente den gleichen Koordinatenraum belegt. Eine Mehrlinienfolge ist einfach, wenn keiner der Innenbereiche ihrer Elemente durch seinen eigenen Innenbereich geschnitten wird.

Die Prädikatfunktion ST_IsSimple verwendet eine Geometrie und gibt 1 (TRUE) zurück, wenn es sich um eine einfache Geometrie handelt und andernfalls 0 (FALSE). Weitere Informationen hierzu finden Sie im Abschnitt „ST_IsSimple" auf Seite 243.

Leer oder nicht leer

Eine Geometrie ist leer, wenn sie keine Punkte enthält. Der Umschlag, die Begrenzung, der Innenbereich und der Außenbereich einer leeren Geometrie sind NULL. Eine leere Geometrie ist immer einfach; sie kann Z-Koordinaten oder Maße enthalten. Leere Linienfolgen und Mehrlinienfolgen haben die Länge 0. Leere Polygone und Multipolygons haben die Fläche 0.

Die Prädikatfunktion ST_IsEmpty verwendet eine Geometrie und gibt 1 (TRUE) zurück, wenn sie leer ist und andernfalls 0 (FALSE). Weitere Informationen hierzu finden Sie im Abschnitt „ST_IsEmpty" auf Seite 239.

Umschlag

Der Umschlag einer Geometrie ist die Begrenzungsgeometrie, die durch die minimalen und maximalen Koordinaten (X,Y) gebildet wird. Die Umschläge der meisten Geometrien bilden ein begrenzendes Rechteck, mit folgenden Ausnahmen:

- der Umschlag eines Punkts ist der Punkt selbst, da seine minimalen und maximalen Koordinaten identisch sind.
- Der Umschlag einer horizontalen oder vertikalen Linienfolge ist eine Linienfolge, die durch die Begrenzung (die Endpunkte) der Quellenlinie gebildet wird.

Die Funktion ST_Envelope verwendet eine Geometrie und gibt eine Begrenzungsgeometrie zurück, die ihren Umschlag darstellt. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Envelope" auf Seite 217.

Dimension

Eine Geometrie kann eine Dimension 0, 1 oder 2 haben. Die Dimensionen sind wie folgt definiert:

0 Hat weder Länge noch Fläche
1 Hat eine Länge
2 Hat eine Fläche

Der Punkt und die Mehrpunkt-Unterklassen haben die Dimension 0. Punkte stellen dimensionale Merkmale dar, die mit einer einzigen Koordinate modelliert werden können; Mehrpunkt-Unterklassen stellen Daten dar, die mit einer Gruppe von nicht miteinander verbundenen Koordinaten modelliert werden müssen.

Die Unterklassen Linienfolge und Mehrlinienfolge haben die Dimension 1. In diesen Unterklassen werden Straßenabschnitte, verzweigte Flußsysteme und andere lineare Merkmale gespeichert.

Die Unterklassen Polygon und Multipolygon haben die Dimension 2. Merkmale, deren Umfang eine definierbare Fläche umfassen, wie beispielsweise Wälder, Flächen oder Wasserflächen, können als Datentyp Polygon oder Multipolygon dargestellt werden.

Die Dimension ist nicht nur als Merkmal der Unterklasse von Bedeutung, sondern spielt auch eine Rolle beim Festlegen der Beziehung zwischen zwei Merkmalen. Die Dimension eines resultierenden Merkmals gibt an, ob die Operation erfolgreich war oder nicht. Der DB2 Spatial Extender untersucht die Dimension der Merkmale, um festzustellen, wie sie verglichen werden sollen. Die Funktion ST_Dimension verwendet eine Geometrie und gibt ihre Dimension als ganze Zahl zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Dimension" auf Seite 211.

Kennung des räumlichen Bezugssystems

Das räumliche Bezugssystem kennzeichnet die Koordinatenumsetzung für jede Geometrie.

Alle in der Datenbank bekannten räumlichen Bezugssysteme können über die Katalogsicht DB2GSE.SPATIAL_REF_SYS aufgerufen werden. Informationen zu dieser Sicht finden Sie in „Kapitel 11. Katalogsichten" auf Seite 115.

Die Funktion ST_SRID verwendet eine Geometrie und gibt ihre räumliche Bezugs-ID als ganze Zahl zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_SRID" auf Seite 277.

Die Funktion ST_Transform ordnet eine Geometrie einem anderen räumlichen Bezugssystem zu als dem momentan zugeordneten. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Transform" auf Seite 282.

Exemplarfähige Geometrien und zugeordnete Funktionen

Dieser Abschnitt umreißt die sechs Unterklassen exemplarfähiger Geometrien und beschreibt die Funktionen, die damit arbeiten. Die Unterklassen sind:

- Punkte
- Linienfolgen
- Polygone
- Mehrpunktangaben
- Mehrlinienfolgen
- Multipolygone

Eine Illustration der Hierarchie, zu der diese Unterklassen gehören sowie der ihnen zugeordneten visuellen Symbole finden Sie in Abb. 9 auf Seite 131.

Punkte

Ein Punkt ist eine Geometrie mit der Dimension 0, die eine einzelne Position in einem Koordinatensystem belegt. Ein Punkt enthält eine X- und eine Y-Koordinate, die diese Position definiert. Er kann auch eine Z-Koordinate und ein Maß enthalten.

Ein Punkt ist einfach und hat die Begrenzung NULL. Punkte werden häufig verwendet, um Merkmale wie Ölquellen, markante Landschaftspunkte oder Erhebungen zu kennzeichnen.

Funktionen, die nur mit der Unterklasse Punkt arbeiten:

ST_Point
Verwendet eine X-Koordinate, die ihr zugeordnete Y-Koordinate und die Kennung des räumlichen Bezugssystems, zu dem diese Koordinaten gehören, und gibt den von diesen Koordinaten definierten Punkt zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Point" auf Seite 268.

ST_CoordDim
Gibt einen Wert zurück, der angibt, welche Koordinaten ein Punkt enthält und ob er ein Maß enthält. Dieser Wert wird als *Koordinatendimension* bezeichnet. Mögliche Koordinatendimensionen sind:

2 Der Punkt besteht aus einer X- und einer Y-Koordinate.

3 Der Punkt besteht aus einer X-, einer Y- und einer Z-Koordinate.

4 Der Punkt besteht aus einer X-, einer Y- und einer Z-Koordinate sowie einem Maß.

Weitere Informationen hierzu finden Sie im Abschnitt „ST_CoordDim" auf Seite 206.

ST_PointFromText
Verwendet eine bekannte OGC-Textdarstellung (WKT) eines Punkts und gibt den Punkt zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_PointFromText" auf Seite 265.

ST_X Gibt den X-Koordinatenwert eines Datentyps ST_Point als Zahl mit doppelter Genauigkeit zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_X" auf Seite 288.

ST_Y Gibt den Y-Koordinatenwert eines Datentyps ST_Point als Zahl mit doppelter Genauigkeit zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Y" auf Seite 289.

Z Gibt den Z-Koordinatenwert eines Datentyps ST_Point als Zahl mit doppelter Genauigkeit zurück. Weitere Informationen hierzu finden Sie im Abschnitt „Z" auf Seite 290.

M Gibt das Maß eines Datentyps ST_Point als Zahl mit doppelter Genauigkeit zurück. Weitere Informationen hierzu finden Sie im Abschnitt „M" auf Seite 183.

Linienfolgen

Eine Linienfolge ist ein eindimensionales Objekt, das als Folge von Punkten gespeichert wird, die einen linear interpolierten Pfad definieren. Die Linienfolge ist einfach, wenn sie ihren Innenbereich nicht schneidet. Die Endpunkte (die Begrenzung) einer geschlossenen Linienfolge belegen den gleichen Punkt im Raum. Eine Linienfolge ist ein Ring, wenn sie geschlossen ist und ihr Innenbereich sich nicht schneidet. Neben den anderen aus der Geometrie der Superklasse übernommenen Merkmalen haben Linienfolgen eine Länge. Linienfolgen werden häufig zur Definition von Straßen, Flüssen oder Stromkabeln verwendet.

Eine einfache Linienfolge, deren Start- und Endpunkt identisch sind, wird als *Ring* bezeichnet.

Die Endpunkte bilden normalerweise die Begrenzung einer Linienfolge, es sei denn, die Linienfolge ist geschlossen; in diesem Fall ist die Begrenzung NULL. Der Innenbereich einer Linienfolge ist der verbundene Pfad zwischen den beiden Endpunkten, es sei denn, die Linienfolge ist geschlossen; in diesem Fall ist der Innenbereich fortlaufend.

Funktionen, die mit Linienfolgen arbeiten:

ST_StartPoint
: Verwendet eine Linienfolge und gibt ihren ersten Punkt zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_StartPoint" auf Seite 278.

ST_EndPoint
: Verwendet eine Linienfolge und gibt ihren letzten Punkt zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Endpoint" auf Seite 216.

ST_PointN
: Verwendet eine Linienfolge und einen Index zum *n*-ten Punkt und gibt diesen Punkt zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_PointN" auf Seite 269.

ST_Length
: Verwendet eine Linienfolge und gibt ihre Länge als Zahl mit doppelter Genauigkeit zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Length" auf Seite 246.

ST_NumPoints

Verwendet eine Linienfolge und gibt die Anzahl ihrer Punkte als ganze Zahl zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_NumPoints" auf Seite 260.

ST_IsRing

Verwendet eine Linienfolge und gibt 1 (TRUE) zurück, wenn die Linienfolge ein Ring ist und andernfalls 0 (FALSE). Weitere Informationen hierzu finden Sie im Abschnitt „ST_IsRing" auf Seite 241.

ST_IsClosed

Verwendet eine Linienfolge und gibt 1 (TRUE) zurück, wenn die Linienfolge geschlossen ist und andernfalls 0 (FALSE). Weitere Informationen hierzu finden Sie im Abschnitt „ST_IsClosed" auf Seite 237.

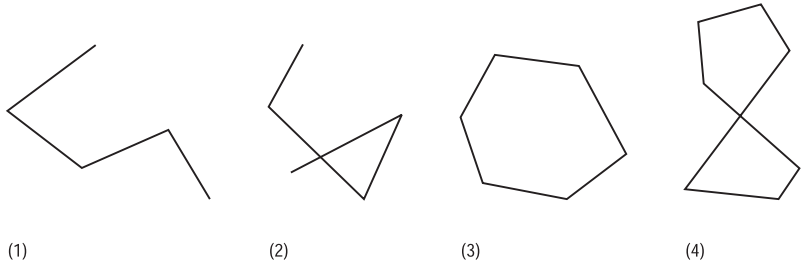


Abbildung 10. Linienfolgeobjekte.

1. Eine einfache, nicht geschlossene Linienfolge.
2. Eine nicht einfache, nicht geschlossene Linienfolge.
3. Eine geschlossene, einfache Linienfolge (ein Ring).
4. Eine geschlossene, nicht einfache Linienfolge (kein Ring).

Polygone

Ein Polygon ist eine zweidimensionale Fläche, die als Folge von Punkten gespeichert wird, die ihren äußeren Begrenzungsring sowie 0 oder mehr innere Ringe definieren. Die Ringe eines Polygons können sich nicht überlappen. Per Definition sind Polygone immer einfach. Meist definieren Polygone Grundstücke, Wasserflächen und andere Merkmale mit einer räumlichen Ausdehnung.

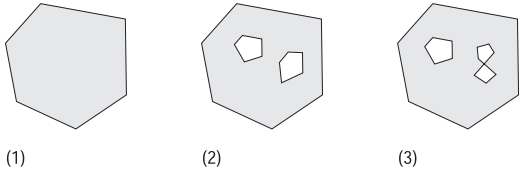


Abbildung 11. Polygone.

1. Ein Polygon, dessen Begrenzung durch einen äußeren Ring definiert ist.
2. Ein Polygon, dessen Begrenzung durch einen äußeren Ring und zwei innere Ringe definiert ist. Die Fläche in den inneren Ringen ist Teil des Außenbereichs der Polygone.
3. Ein zulässiges Polygon, da sich die Ringe an einem einzigen Tangentenpunkt berühren.

Der Außenbereich und alle inneren Ringe definieren die Begrenzung eines Polygons, und der zwischen den Ringen eingeschlossene Bereich definiert den Innenbereich des Polygons. Die Ringe eines Polygons können sich in einem Tangentenpunkt berühren, aber niemals schneiden. Neben den anderen aus der Geometrie der Superklasse übernommenen Merkmalen haben Polygone eine Fläche.

Funktionen, die mit Polygonen arbeiten:

ST_Area
: Verwendet ein Polygon und gibt seine Fläche als Zahl mit doppelter Genauigkeit zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Area" auf Seite 193.

ST_ExteriorRing
: Verwendet ein Polygon und gibt seinen äußeren Ring als Linienfolge zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_ExteriorRing" auf Seite 220.

ST_NumInteriorRing
Verwendet ein Polygon und gibt die Anzahl der darin enthaltenen inneren Ringe zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_NumInteriorRing" auf Seite 259.

ST_InteriorRingN
Verwendet ein Polygon und einen Index und gibt den *n*-ten inneren Ring als Linienfolge zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_InteriorRingN" auf Seite 229.

ST_Centroid
Verwendet ein Polygon und gibt einen Punkt zurück, der den Mittelpunkt der Ausdehnung des Polygons darstellt. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Centroid" auf Seite 201.

ST_PointOnSurface
Verwendet ein Polygon und gibt einen Punkt zurück, der garantiert in der Fläche des Polygons liegt. Weitere Informationen hierzu finden Sie im Abschnitt „ST_PointOnSurface" auf Seite 270.

ST_Perimeter
Verwendet ein Polygon und gibt den Umfang seiner Fläche zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Perimeter" auf Seite 264.

Mehrpunktangaben

Eine Mehrpunktangabe ist eine Gruppe von Punkten; sie hat wie ihre Elemente die Dimension 0. Eine Mehrpunktangabe ist einfach, wenn keine ihrer Elemente den gleichen Koordinatenraum belegen. Die Begrenzung einer Mehrpunktangabe ist NULL. Mehrpunktangaben können zur Definition von Phänomenen wie Funkverteilungsmustern oder den Punkten eines Epidemieausbruchs verwendet werden.

Funktionen, die mit Mehrpunktangaben arbeiten:

ST_NumGeometries
Verwendet eine homogene Gruppe und gibt die Anzahl der darin enthaltenen Basisgeometrieelemente zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_NumGeometries" auf Seite 258.

ST_GeometryN
Verwendet eine homogene Gruppe und einen Index und gibt die n-te Basisgeometrie zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_GeometryN" auf Seite 226.

Mehrlinienfolgen

Eine Mehrlinienfolge ist eine Gruppe von Linienfolgen. Mehrlinienfolgen sind einfach, wenn sie sich lediglich in den Endpunkten der Linienfolgen berühren, aber nicht schneiden. Mehrlinienfolgen sind nicht einfach, wenn sich die Innenbereiche der Linienfolgen schneiden.

Die Begrenzung einer Mehrlinienfolge sind die sich nicht berührenden Endpunkte der Linienfolgenelemente. Die Mehrlinienfolge gilt als geschlossen, wenn alle ihre Linienfolgen geschlossen sind. Die Begrenzung einer Mehrlinienfolge ist NULL, wenn sich alle Endpunkte aller ihrer Elemente berühren. Neben den anderen aus der Geometrie der Superklasse übernommenen Merkmalen haben Mehrlinienfolgen eine Länge. Mehrlinienfolgen werden zur Definition von Bach- der Straßennetzen verwendet.

Funktionen, die mit Mehrlinienfolgen arbeiten:

ST_Length
Verwendet eine Mehrlinienfolge und gibt die kumulierte Länge all ihrer Linienfolgen als Zahl mit doppelter Genauigkeit zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Length" auf Seite 246.

ST_IsClosed
Verwendet eine Mehrlinienfolge und gibt 1 (TRUE) zurück, wenn die Mehrlinienfolge geschlossen ist und andernfalls 0 (FALSE). Weitere Informationen hierzu finden Sie im Abschnitt „ST_IsClosed" auf Seite 237.

ST_NumGeometries
Verwendet eine homogene Gruppe und gibt die Anzahl der darin enthaltenen Basisgeometrieelemente zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_NumGeometries" auf Seite 258.

ST_GeometryN
Verwendet eine homogene Gruppe und einen Index und gibt die n-te Basisgeometrie zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_GeometryN" auf Seite 226.

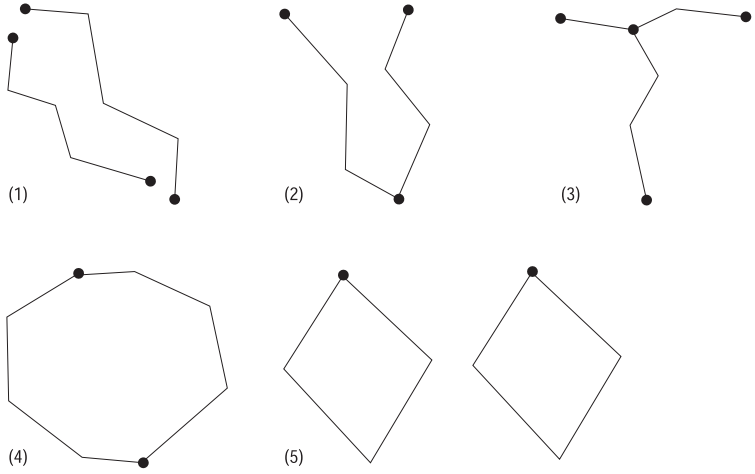
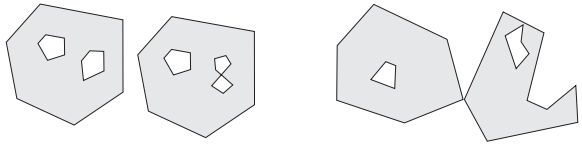


Abbildung 12. Mehrlinienfolgen.

1. Eine einfache Mehrlinienfolge, deren Begrenzung durch die vier Endpunkte ihrer beiden Linienfolgen definiert ist.
2. Eine einfache Mehrlinienfolge, da sich nur die Endpunkte der Linienfolgen berühren. Die Begrenzung ist durch die beiden sich nicht berührenden Endpunkt definiert.
3. Eine nicht einfache Mehrlinienfolge, da der Innenbereich einer ihrer Linienfolgen geschnitten wird. Die Begrenzung dieser Mehrlinienfolge ist durch die vier Endpunkte einschließlich des Verbindungspunkts definiert.
4. Eine einfache, nicht geschlossene Mehrlinienfolge. Sie ist nicht geschlossen, da ihre Linienfolgen nicht geschlossen sind. Sie ist einfach, weil keiner der Innenbereiche der darin enthaltenen Linienfolgen geschnitten wird.
5. Eine einfache geschlossene Mehrlinienfolge. Sie ist geschlossen, weil alle ihre Elemente geschlossen sind. Sie ist einfach, weil keines ihrer Elemente im Innenbereich geschnitten wird.

Multipolygone

Die Begrenzung eines Multipolygons ist die kumulierte Länge der äußeren und inneren Ringe seiner Elemente. Der Innenbereich eines Multipolygons ist definiert als die kumulierten Innenbereiche seiner Elementpolygone. Die Begrenzung der Elemente eines Multipolygons können sich nur in einem Tangentenpunkt berühren. Neben den weiteren aus der Superklassen-Geometrie übernommenen Merkmalen haben Multipolygone eine Fläche. Multipolygone definieren Merkmale wie beispielsweise ein Waldstück oder ein nicht zusammenhängendes Stück Land wie beispielsweise eine Inselkette.



(1) (2)

Abbildung 13. Multipolygone.

1. Ein Multipolygon mit zwei Polygonelementen. Die Begrenzung ist durch die beiden äußeren Ringe und die drei inneren Ringe definiert.
2. Ein Multipolygon mit zwei Polygonelementen. Die Begrenzung ist durch die beiden äußeren Ringe und die beiden inneren Ringe definiert. Die beiden Polygonelemente berühren sich an einem Tangentenpunkt.

Funktionen, die mit Multipolygonen arbeiten:

ST_Area
Verwendet eine Multipolygon und gibt die kumulierte Fläche seiner Polygonelemente als Zahl mit doppelter Genauigkeit zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Area" auf Seite 193.

ST_Centroid
Verwendet ein Multipolygon und gibt einen Punkt zurück, der seinen geometrisch gewichteten Mittelpunkt darstellt. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Centroid" auf Seite 201.

ST_NumGeometries
Verwendet eine homogene Gruppe und gibt die Anzahl der darin enthaltenen Basisgeometrieelemente zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_NumGeometries" auf Seite 258.

ST_GeometryN
Verwendet eine homogene Gruppe und einen Index und gibt die n-te Basisgeometrie zurück. Weitere Informationen hierzu finden Sie im Abschnitt „ST_GeometryN" auf Seite 226.

Funktionen, die Beziehungen und Vergleiche zeigen, Geometrien generieren und Werteformate umsetzen

In den vorangegangenen Abschnitten wurden drei Kategorien räumlicher Funktionen vorgestellt:

- Funktionen zu den Merkmalen der Geometrien
- Funktionen zu den spezifischen Geometrien

In diesem Abschnitt werden drei weitere Kategorien vorgestellt:

- Funktionen, die Möglichkeiten der Beziehung und des Vergleichs geographischer Merkmale festlegen
- Funktionen, die neue Geometrien generieren
- Funktionen, die die Werte einer Geometrie in ein Format umsetzen, das importiert oder exportiert werden kann

Funktionen, die Beziehungen oder Vergleiche zwischen geographischen Merkmalen zeigen

Verschiedene räumliche Funktionen geben Informationen zurück über die Beziehung oder den Vergleich zwischen verschiedenen geographischen Merkmalen. Die meisten dieser Funktionen oder *Prädikate* sind Boolesche Funktionen. In diesem Abschnitt werden die Prädikate im allgemeinen und anschließend die Funktionen im einzelnen beschrieben.

Prädikatfunktionen

Prädikatfunktionen geben 1 (TRUE) zurück, wenn ein Vergleich die Kriterien der Funktion erfüllt oder 0 (FALSE), wenn der Vergleich fehlschlägt. Prädikate, die einen Test auf eine räumliche Beziehung durchführen, vergleichen Paare von Geometrien, die einen unterschiedlichen Typ oder unterschiedliche Dimensionen haben können.

Prädikate vergleichen die X- und Y-Koordinaten der übergebenen Geometrien. Die Z-Koordinaten und das Maß (sofern vorhanden) werden nicht berücksichtigt. Auf diese Weise können Geometrien, die Z-Koordinaten oder ein Maß haben, mit anderen Geometrien ohne diese Merkmale verglichen werden.

Das *Dimensionally Extended 9 Intersection Model (DE-9IM)* [1] ist ein mathematischer Ansatz, der die paarweise räumliche Beziehung zwischen Geometrien mit verschiedenen Typen und Dimensionen definiert. Dieses Modell drückt die räumliche Beziehung zwischen allen Typen von Geometrien als paarweise Schnittmengen ihrer Innenbereiche, Begrenzungen und Außenbereiche unter Berücksichtigung der Dimension der resultierenden Schnittmengen aus.

Die gegebenen Geometrien *a* und *b*: I(*a*), B(*a*) und E(*a*) stellen Innenbereich, die Begrenzung und den Außenbereich von *a* dar. I(b), B(b) und E(b) stellen den Innenbereich, die Begrenzung und den Außenbereich von *b* dar. Die Schnittmengen von I(a), B(a) und E(a) mit I(b), B(b) und E(b) ergeben eine 3-mal-3-Matrix. Jede Schnittmenge kann Geometrien verschiedener Dimensionen ergeben. Die Schnittmenge der Begrenzungen zweier Polygone besteht beispielsweise aus einem Punkt und einer Linienfolge; in diesem Fall gibt die dim-Funktion die maximale Dimension 1 zurück.

Die dim-Funktion gibt einen Wert 1, 0, 1 oder 2 zurück. Die 1 entspricht dem Null-Set oder dim(null); dieses Ergebnis wird zurückgegeben, wenn keine Schnittmenge gefunden wurde.

	Innenbereich	**Begrenzung**	**Außenbereich**
Innenbereich	dim(I(a) ∩ I(b))	dim(I(a) ∩ B(b))	dim(I(a) ∩ E(b))
Begrenzung	dim(B(a) ∩ I(b))	dim(B(a) ∩ B(b))	dim(B(a) ∩ E(b))
Außenbereich	dim(E(a) ∩ I(b))	dim(E(a) ∩ B(b))	dim(E(a) ∩ E(b))

Sie können das Ergebnis der räumlichen Beziehungsprädikate verstehen oder überprüfen, indem Sie die Ergebnisse des Prädikats mit einer Mustermatrix vergleichen, die die akzeptablen Werte des DE-9IM darstellt.

1. Das DE-91M wurde von Clementini und Felice entwickelt, die das 9 Intersection Model von Egenhofer und Herring dimensional erweiterten. DE-91M ist das Ergebnis der Zusammenarbeit der vier Autoren Clementini, Eliseo, Di Felice und van Osstrom. Diese Autoren haben das Modell in "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, veröffentlicht. Das 9 Intersection Modell vom Springer-Verlag Singapore (1993) Egenhofer M.J. und Herring, J. wurde in "Categorizing binary topological relationships between regions, lines, and points in geographic databases," *Tech. Report, Department of Surveying Engineering*, University of Maine, Orono, ME 1991, veröffentlicht.

Die Mustermatrix enthält die akzeptablen Werte für alle Schnittmengen-Matrixzellen. Die möglichen Musterwerte sind:

T Es muß eine Schnittmenge vorhanden sein, dim = 0, 1 oder 2.

F Es darf keine Schnittmenge vorhanden sein, dim = -1.

***** Es spielt keine Rolle, ob eine Schnittmenge vorhanden ist, dim = -1, 0, 1 oder 2.

0 Es muß eine Schnittmenge vorhanden sein, und ihre maximale Dimension muß 0 sein, dim = 0.

1 Es muß eine Schnittmenge vorhanden sein, und ihre maximale Dimension muß 1 sein, dim = 1.

2 Es muß eine Schnittmenge vorhanden sein, und ihre maximale Dimension muß 2 sein, dim = 2.

Die folgende Mustermatrix für das Prädikat ST_Within enthält beispielsweise die Werte T, F und *.

Tabelle 40. Matrix für ST_Within. Die Mustermatrix für das Prädikat ST_Within für Geometriekombinationen.

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	T	*	F
	Begrenzung	*	*	F
	Außenbereich	*	*	*

Das Prädikat ST_Within gibt TRUE zurück, wenn sich die Innenbereiche der beiden Geometrien schneiden und wenn der Innenbereich und die Begrenzung von *a* sich nicht mit dem Außenbereich von *b* schneidet. Alle weitere Bedingungen sind nicht von Bedeutung.

Jedes Prädikat hat mindestens eine Mustermatrix; einige erfordern jedoch auch mehrere Matrizen, um die Beziehungen zwischen verschiedenen Kombinationen von Geometrietypen zu beschreiben.

ST_Equals

ST_Equals gibt 1 (TRUE) zurück, wenn zwei Geometrien des gleichen Typs identische Koordinaten X,Y haben.

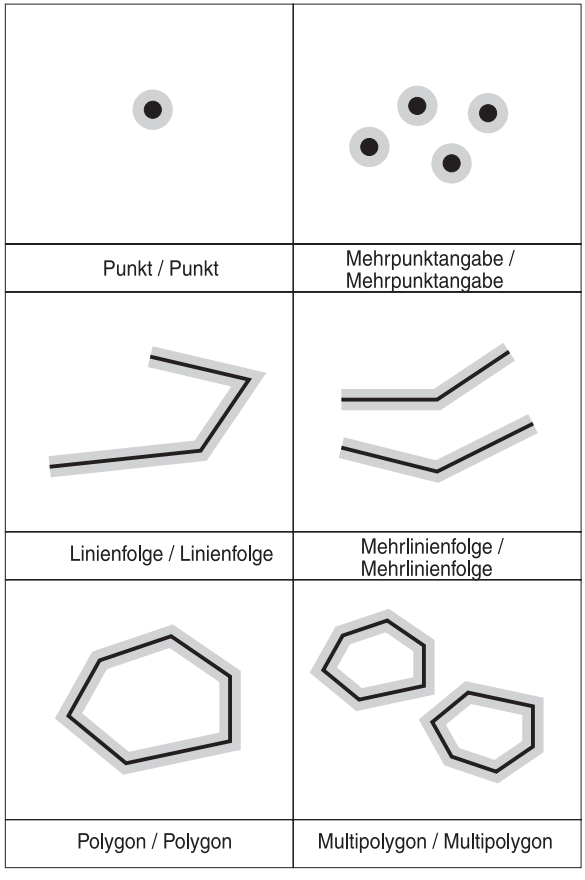


Abbildung 14. ST_Equals. Geometrien sind gleich, wenn sie übereinstimmende Koordinaten X,Y haben.

Tabelle 41. Matrix für Gleichheit. Die DE-9IM Mustermatrix für die Gleichheit stellt sicher, daß sich die Innenbereiche schneiden und daß kein Teil des Innenbereichs oder der Begrenzung den Außenbereich des anderen schneidet.

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	T	*	F
	Begrenzung	*	*	F
	Außenbereich	F	F	*

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Equals" auf Seite 219.

ST_OrderingEquals

ST_OrderingEquals vergleicht zwei Geometrien und gibt 1 (TRUE) zurück, wenn die Geometrien gleich sind und die Koordinaten die gleiche Reihenfolge haben; andernfalls wird 0 (FALSE) zurückgegeben. Weitere Informationen hierzu finden Sie im Abschnitt „ST_OrderingEquals" auf Seite 261.

ST_Disjoint

ST_Disjoint gibt 1 (TRUE) zurück, wenn die Schnittmenge der beiden Geometrien eine leere Gruppe ist.

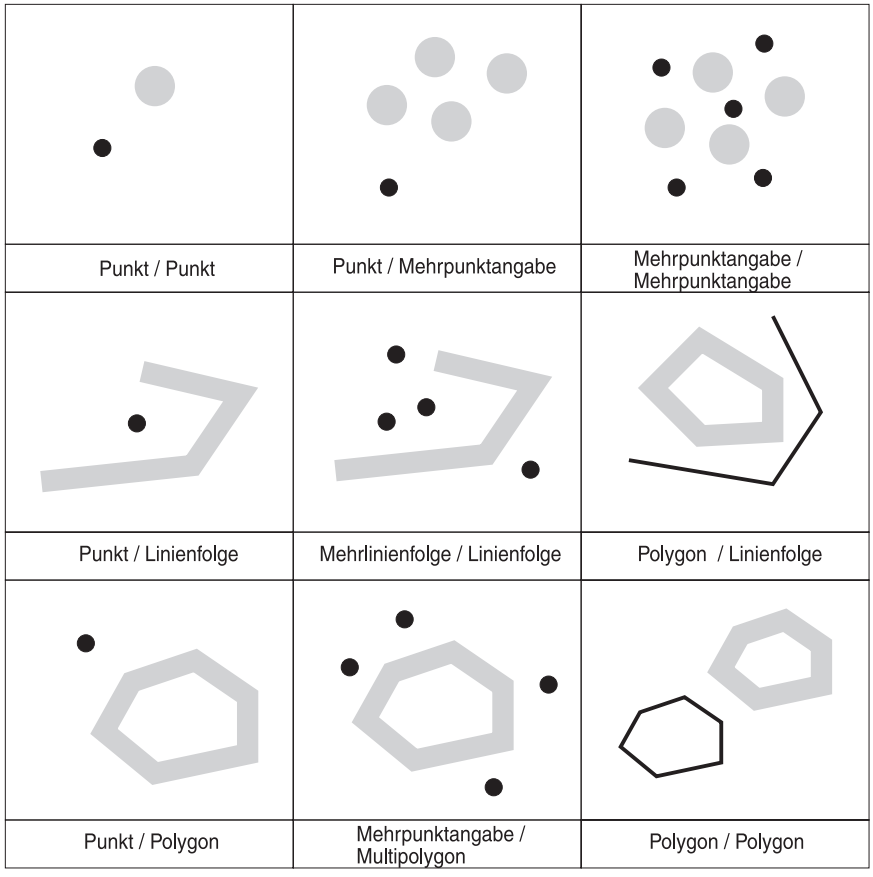


Abbildung 15. ST_Disjoint. Geometrien sind nicht verbunden, wenn sie einander nicht schneiden.

Tabelle 42. Matrix für ST_Disjoint. Die Mustermatrix des Prädikats ST_Disjoint gibt nur an, daß sich weder die Innenbereiche noch die Begrenzungen der Geometrien schneiden.

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	F	F	*
	Begrenzung	F	F	*
	Außenbereich	*	*	*

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Disjoint" auf Seite 213.

ST_Intersects

ST_Intersects gibt 1 (TRUE) zurück, wenn die Schnittmenge nicht zu einer leeren Gruppe führt. ST_Intersects gibt genau das entgegengesetzte Ergebnis von ST_Disjoint zurück.

Das Prädikat ST_Intersects gibt TRUE zurück, wenn die Bedingung einer der folgenden Mustermatrizen TRUE zurückgibt.

Tabelle 43. Matrix für ST_Intersects (1). Das Prädikat ST_Intersects gibt TRUE zurück, wenn sich die Innenbereiche der beiden Geometrien schneiden.

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	T	*	*
	Begrenzung	*	*	*
	Außenbereich	*	*	*

Tabelle 44. Matrix für ST_Intersects (2). Das Prädikat ST_Intersects gibt TRUE zurück, wenn die Begrenzung der ersten Geometrie die Begrenzung der zweiten Geometrie schneidet.

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	*	T	*
	Begrenzung	*	*	*
	Außenbereich	*	*	*

Tabelle 45. Matrix für ST_Intersects (3). Das Prädikat ST_Intersects gibt TRUE zurück, wenn die Begrenzung der ersten Geometrie den Innenbereich der zweiten Geometrie schneidet.

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	*	*	*
	Begrenzung	T	*	*
	Außenbereich	*	*	*

Tabelle 46. Matrix für ST_Intersects (4). Das Prädikat ST_Intersects gibt TRUE zurück, wenn sich die Begrenzungen der beiden Geometrien schneiden.

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	*	*	*
	Begrenzung	*	T	*
	Außenbereich	*	*	*

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Intersects" auf Seite 236.

EnvelopesIntersect

Diese Funktion gibt 1 (TRUE) zurück, wenn sich die Umschläge der beiden Geometrien schneiden. Es ist eine hilfreiche Funktion, die ST_Intersects (`ST_Envelope(g1),ST_Envelope(g2)`) effizient implementiert. Weitere Informationen hierzu finden Sie im Abschnitt „EnvelopesIntersect" auf Seite 173.

ST_Touches

ST_Touches gibt 1 (TRUE) zurück, wenn keiner der gemeinsamen Punkte der beiden Geometrien den Innenbereich beider Geometrien schneidet. Mindestens eine der Geometrien muß eine Linienfolge, ein Polygon, eine Mehrlinienfolge oder ein Multipolygon sein.

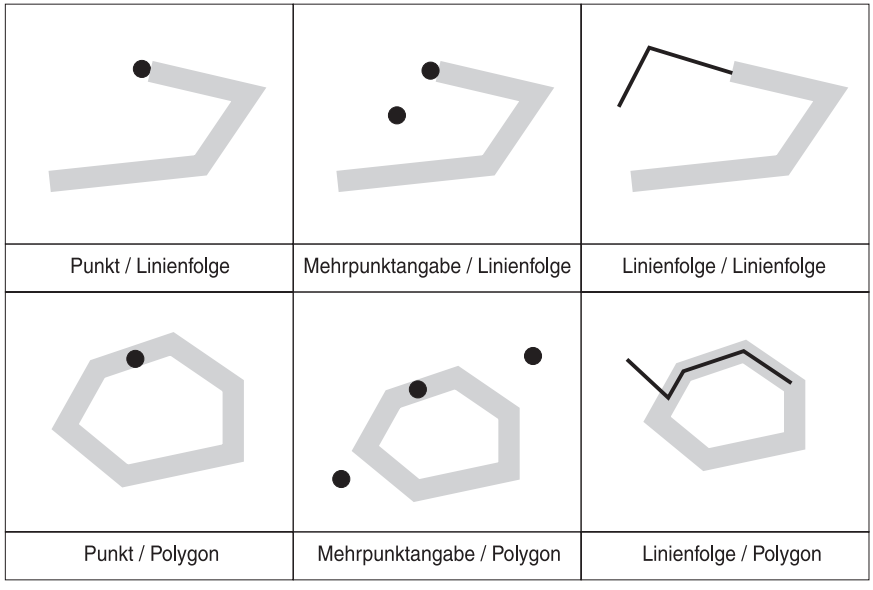


Abbildung 16. ST_Touches

Die Mustermatrizen zeigen, daß das Prädikat ST_Touches TRUE zurückgibt, wenn sich die Innenbereiche der Geometrie nicht schneiden und die Begrenzungen einer der beiden Geometrien den Innenbereich oder die Begrenzung der anderen Geometrie schneiden.

Tabelle 47. Matrix für ST_Touches (1)

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	F	T	*
	Begrenzung	*	*	*
	Außenbereich	*	*	*

Tabelle 48. Matrix für ST_Touches (2)

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	F	*	*
	Begrenzung	T	*	*
	Außenbereich	*	*	*

Tabelle 49. Matrix für ST_Touches (3)

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	F	*	*
	Begrenzung	*	T	*
	Außenbereich	*	*	*

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Touches" auf Seite 281.

ST_Overlaps

ST_Overlaps vergleicht zwei Geometrien der gleichen Dimension. Sie gibt 1 (TRUE) zurück, wenn die Schnittmenge eine Geometrie ergibt, die sich von beiden Geometrien unterscheidet, aber die gleiche Dimension hat.

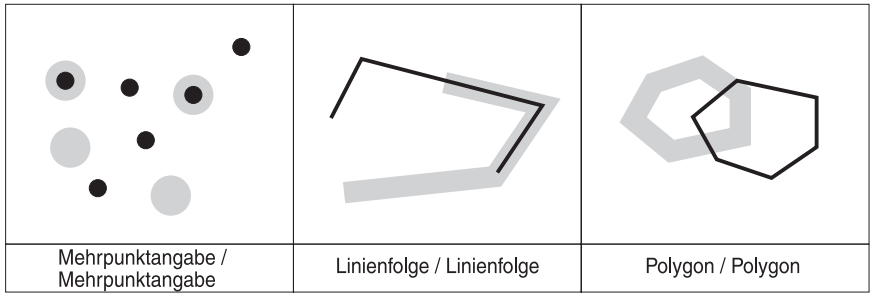


Abbildung 17. ST_Overlaps

Die Mustermatrix in Tabelle 50 auf Seite 154 gilt für die Überlagerungen Polygon/Polygon, Mehrpunktangabe/Mehrpunktangabe und Multipolygon/Multipolygon. Für diese Kombinationen gibt das Überlagerungsprädikat TRUE zurück, wenn der Innenbereich beider Geometrien den Innenbereich und Außenbereich der anderen Geometrie schneidet.

Tabelle 50. Matrix für ST_Overlaps (1)

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	T	*	T
	Begrenzung	*	*	*
	Außenbereich	T	*	*

Die Mustermatrix in Tabelle 51 gilt für die Überlagerungen Linienfolge/Linienfolge und Mehrlinienfolge/Mehrlinienfolge. In diesem Fall muß die Schnittmenge der Geometrien eine Geometrie mit der Dimension 1 (andere Linienfolge) ergeben. Wenn die Dimension der Schnittmenge der Innenbereiche 1 ist, gibt das Prädikat ST_Overlaps FALSE zurück; das Prädikat ST_Crosses gibt jedoch TRUE zurück.

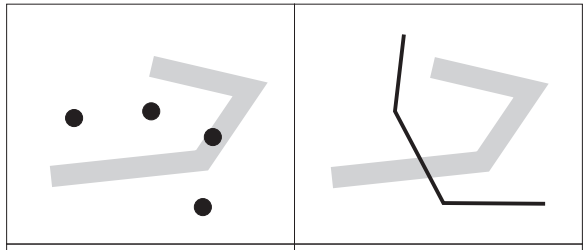
Tabelle 51. Matrix für ST_Overlaps (2)

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	1	*	T
	Begrenzung	*	*	*
	Außenbereich	T	*	*

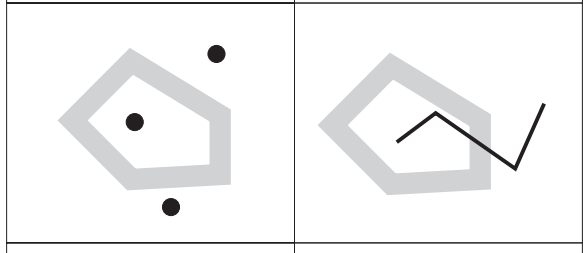
Weitere Informationen hierzu finden Sie im Abschnitt „ST_Overlaps" auf Seite 262.

ST_Crosses

ST_Crosses gibt 1 (TRUE) zurück, wenn die Schnittmenge eine Geometrie ergibt, deren Dimension um eins niedriger ist als die maximale Dimension der beiden Quellenobjekte und wenn die Schnittmenge im Innenbereich beider Quellengeometrien liegt. ST_Crosses gibt nur 1 (TRUE) zurück für Vergleiche Mehrpunktangabe/Polygon, Mehrpunktangabe/Linienfolge, Linienfolge/Linienfolge, Linienfolge/Polygon und Linienfolge/Multipolygon.



Mehrpunktangabe / Linienfolge	Linienfolge / Linienfolge



Mehrpunktangabe / Polygon	Linienfolge / Polygon

Die Mustermatrix in Tabelle 52 gilt für Mehrpunktangabe/Linienfolge, Mehrpunktangabe/Mehrlinienfolge, Mehrpunktangabe/Polygon, Mehrpunktangabe/Multipolygon, Linienfolge/Polygon und Linienfolge/Multipolygon. Die Matrix gibt an, daß sich die Innenbereiche schneiden müssen und daß der Innenbereich der primären Geometrie (Geometrie *a*) den Außenbereich der sekundären Geometrie (Geometrie *b*) schneiden muß.

Tabelle 52. Matrix für ST_Crosses (1)

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	T	*	T
	Begrenzung	*	*	*
	Außenbereich	*	*	*

Die Mustermatrix in Tabelle 53 auf Seite 156 gilt für die Überlagerungen Linienfolge/Linienfolge, Linienfolge/Mehrlinienfolge und Mehrlinienfolge/Mehrlinienfolge. Die Matrix gibt an, daß die Dimension der Schnittmenge der Innenbereiche 0 (Berührung in einem Punkt) sein muß. Wenn die Dimension dieser Schnittmenge 1 ist (Schnitt in einer Linienfolge), gibt das Prädikat ST_Crosses FALSE zurück; das Prädikat ST_Overlaps gibt jedoch TRUE zurück.

Tabelle 53. Matrix für ST_Crosses (2)

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	0	*	*
	Begrenzung	*	*	*
	Außenbereich	*	*	*

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Crosses" auf Seite 208.

ST_Within

ST_Within gibt 1 (TRUE) zurück, wenn die Geometrie vollständig in der zweiten Geometrie enthalten ist. ST_Within gibt genau das entgegengesetzte Ergebnis von ST_Contains zurück.

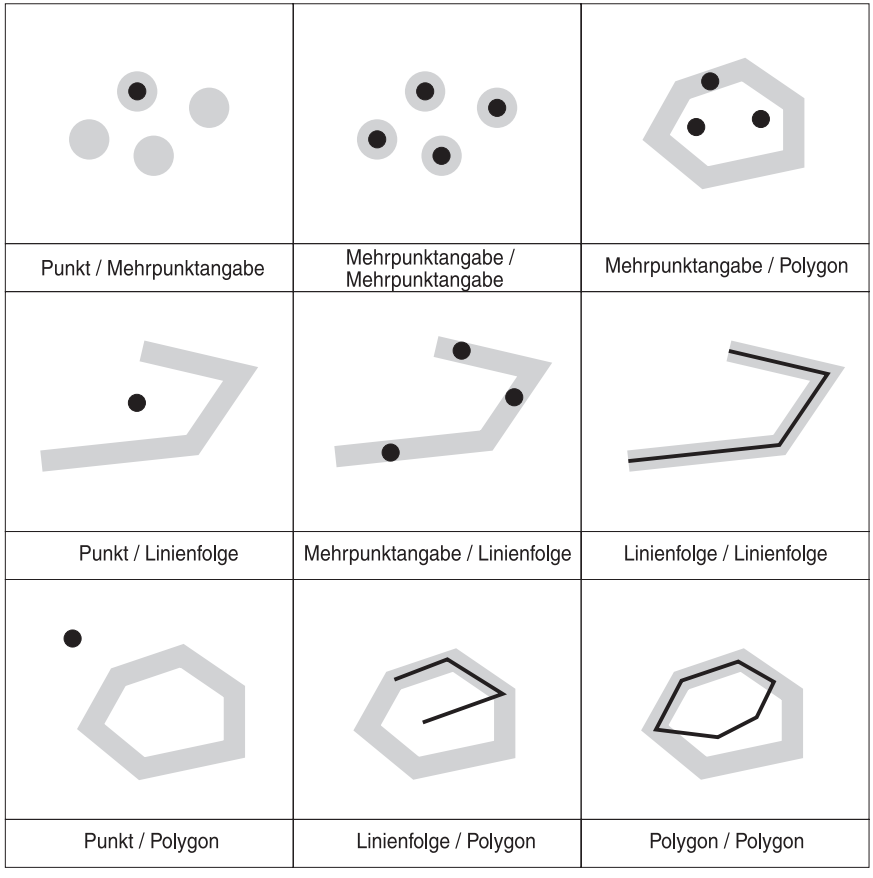


Abbildung 18. ST_Within

Die Mustermatrix von ST_Within gibt an, daß sich die Innenbereiche der beiden Geometrien schneiden müssen und daß der Innenbereich und die Begrenzung der primären Geometrie (Geometrie *a*) den Außenbereich der sekundären Geometrie (Geometrie *b*) nicht schneiden dürfen.

Tabelle 54. Matrix für ST_Within

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	T	*	F
	Begrenzung	*	*	F
	Außenbereich	*	*	*

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Within" auf Seite 284.

ST_Contains

ST_Contains gibt 1 (TRUE) zurück, wenn die zweite Geometrie vollständig in der ersten Geometrie enthalten ist. ST_Contains gibt genau das entgegengesetzte Ergebnis von ST_Within zurück.

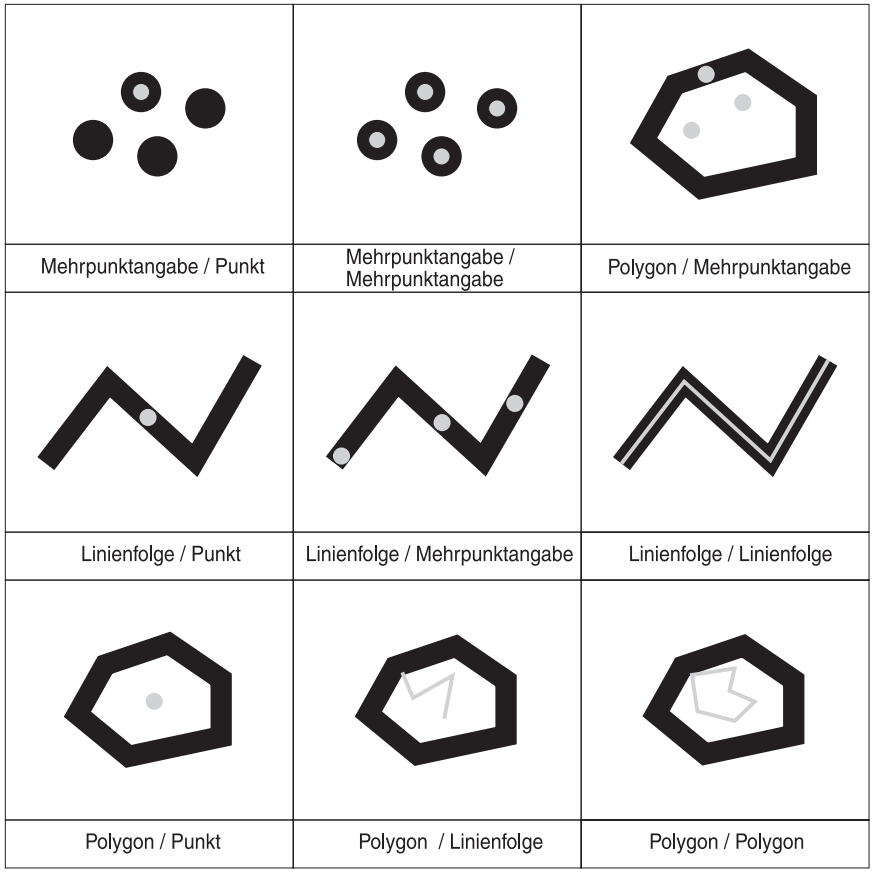


Abbildung 19. ST_Contains

Die Mustermatrix von ST_Contains gibt an, daß sich die Innenbereiche der beiden Geometrien schneiden müssen und daß der Innenbereich und die Begrenzung der zweiten Geometrie (Geometrie *b*) den Innenbereich der primären Geometrie (Geometrie *a*) nicht schneiden dürfen.

Tabelle 55. Matrix für ST_Contains

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
a	**Innenbereich**	T	*	*
	Begrenzung	*	*	*
	Außenbereich	F	F	*

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Contains" auf Seite 202.

ST_Relate

Die Funktion ST_Relate vergleicht zwei Geometrien und gibt 1 (TRUE) zurück, wenn die Geometrien die von der Mustermatrix-Zeichenfolge DE-91M definierten Bedingungen erfüllen; andernfalls wird 0 (FALSE) zurückgegeben. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Relate" auf Seite 275.

ST_Distance

Die Funktion ST_Distance liefert die Mindestentfernung, die zwei nicht verbundene Merkmale trennt. Wenn die Merkmale nicht getrennt sind, meldet die Funktion den Mindestabstand 0.

ST_Distance könnte beispielsweise die kürzeste Entfernung angeben, mit der ein Flugzeug die Strecken zwischen zwei Standorten überbrücken könnte. Abb. 20 verdeutlicht diese Angabe.

Abbildung 20. Mindestentfernung zwischen zwei Städten. ST_Distance kann die Koordinaten der Standorte von Los Angeles und Chicago als Eingabe verwenden und einen Wert zurückgeben, der die Mindestentfernung zwischen diesen beiden Standorten angibt.

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Distance" auf Seite 215.

Funktionen, die neue Geometrien aus vorhandenen generieren

Der DB2 Spatial Extender bietet Prädikate und Umsetzungsfunktionen, die neue Geometrien aus vorhandenen generieren.

ST_Intersection

Die Funktion ST_Intersection gibt die Schnittmenge der beiden Geometrien zurück. Die Schnittmenge wird immer als Gruppe zurückgegeben, die die Mindestdimension der Quellengeometrien darstellt. Für eine Linienfolge, die ein Polygon schneidet, gibt die Funktion ST_Intersection beispielsweise eine Mehrlinienfolge zurück, die aus einem Abschnitt dieser Linienfolge besteht, der zum Innenbereich und zur Begrenzung des Polygons gehört. Die Mehrlinienfolge enthält mehrere Linienfolgen, wenn die Quellenlinienfolge das Polygon mit zwei oder mehr nicht zusammenhängenden Segmenten schnei-

det. Wenn sich die Geometrien nicht schneiden oder wenn die Schnittmenge eine Dimension ergibt, die kleiner ist als die der Quellengeometrien, wird eine leere Geometrie zurückgegeben.

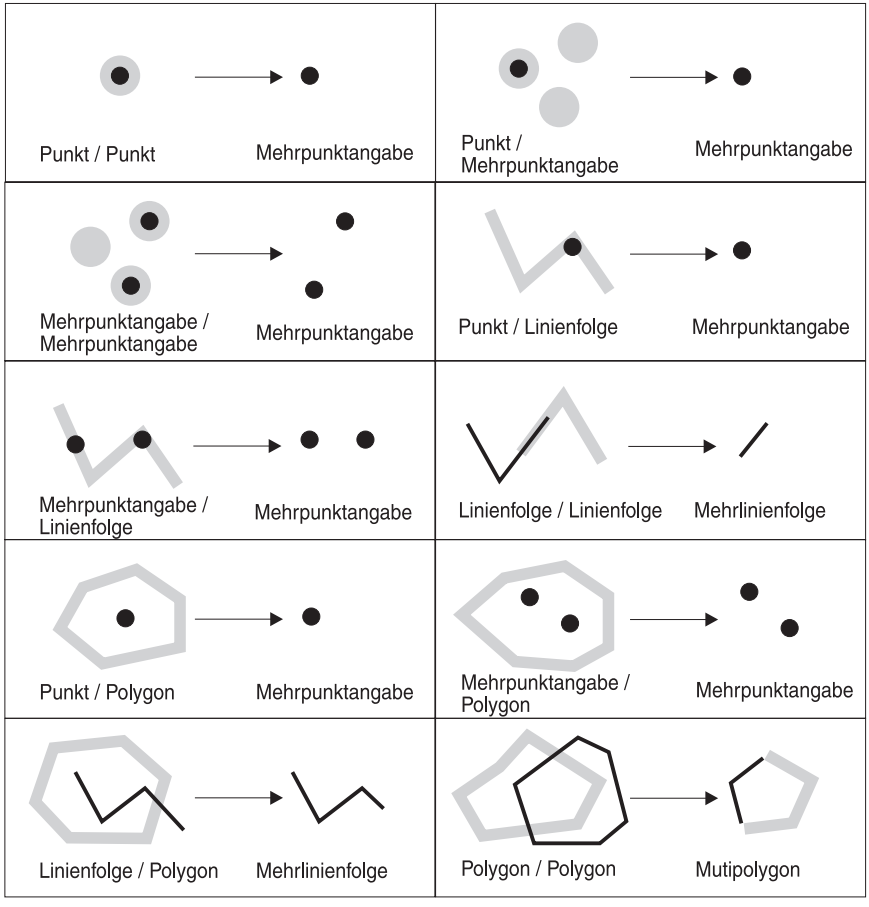


Abbildung 21. ST_Intersection. Beispiele zur Funktion ST_Intersection.

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Intersection" auf Seite 234.

ST_Difference

Die Funktion ST_Difference gibt den Teil der primären Geometrie zurück, der nicht von der zweiten Geometrie geschnitten wird. Dies entspricht der logischen räumlichen Verknüpfung AND NOT. Die Funktion ST_Difference arbeitet nur mit Geometrien gleicher Dimensionen und gibt eine Gruppe zurück, die die gleiche Dimension hat wie die Quellengeometrien. Falls die Quellengeometrien gleich sind, wird eine leere Geometrie zurückgegeben.

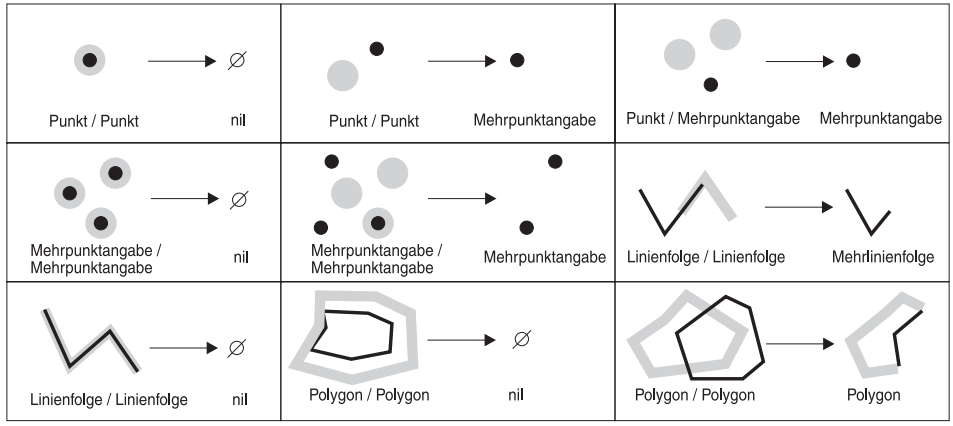


Abbildung 22. ST_Difference

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Difference" auf Seite 210 .

ST_Union

Die Funktion ST_Union gibt die Verknüpfungsmenge der beiden Geometrien zurück. Dies entspricht der logischen räumlichen Funktion OR. Die Quellengeometrien müssen die gleiche Dimension haben. ST_Union gibt das Ergebnis immer als Gruppe zurück.

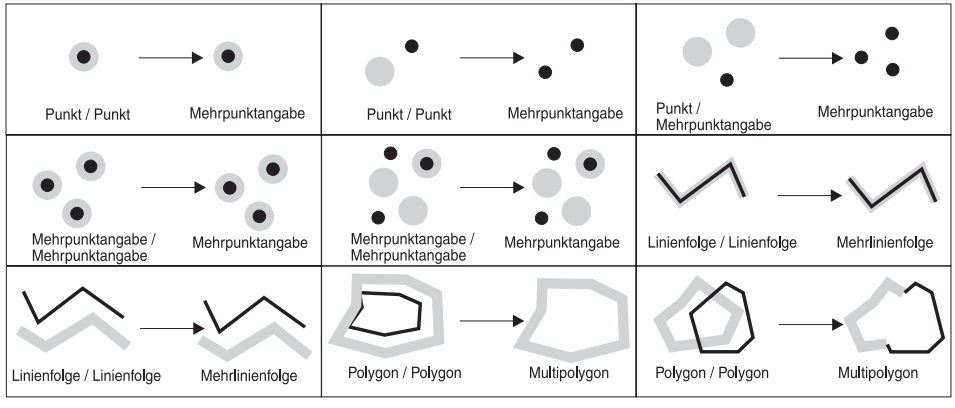


Abbildung 23. ST_Union

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Union" auf Seite 283.

ST_Buffer

Die Funktion ST_Buffer generiert eine Geometrie durch Einkreisen einer Geometrie mit einem angegebenen Abstand. Ein Polygon ergibt sich, wenn eine

primäre Geometrie gepuffert wird oder wenn die Elemente einer Gruppe eng genug beieinander liegen, so daß sich alle Polygonpuffer überlappen. Wenn genügend Abstand zwischen den Elementen einer gepufferten Gruppe liegt, ergeben sich einzelne Pufferpolygone; in diesem Fall gibt die Funktion ST_Buffer ein Multipolygon zurück.

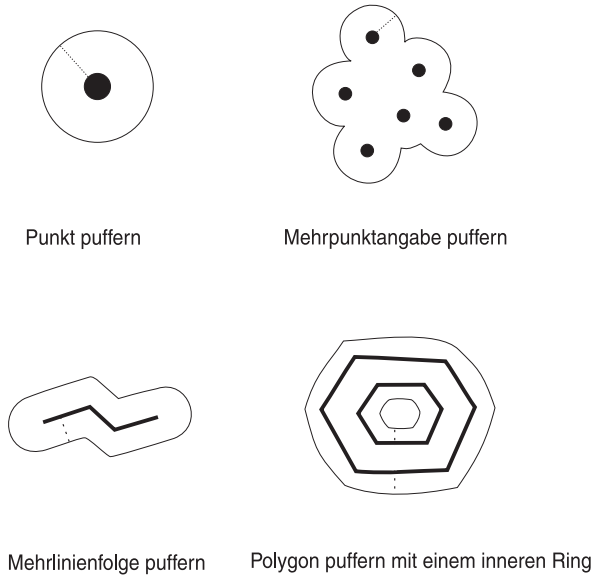


Abbildung 24. ST_Buffer

Die Funktion ST_Buffer verwendet positive und negative Abstände; für negative Puffer werden jedoch nur Geometrien mit der Dimension 2 (Polygone und Multipolygone) angewendet. Der absolute Wert des Pufferabstands wird verwendet, wenn die Dimension der Quellengeometrie weniger als 2 (alle Geometrien außer Polygon oder Multipolygon) beträgt.

Im allgemeinen generieren positive Pufferabstände für äußere Ringe Polygonringe, die sich außerhalb der Mitte der Quellengeometrie befinden; negative Pufferabstände generieren Polygon- oder Multipolygonringe zur Mitte hin. Für innere Ringe eines Polygons oder Multipolygons generiert ein positiver Pufferabstand einen Pufferring zur Mitte hin, und ein negativer Pufferabstand generiert einen Pufferring außerhalb der Mitte.

Der Pufferungsprozeß mischt überlappende Polygone. Negative Abstände größer als die Hälfte der maximalen Breite des Innenbereichs eines Polygons ergeben eine leere Geometrie.

Weitere Informationen hierzu finden Sie im Abschnitt „ST_Buffer" auf Seite 199.

LocateAlong

Für Geometrien mit Maßen kann die Position eines bestimmten Maßes mit der Funktion LocateAlong ermittelt werden. LocateAlong gibt die Position als Mehrpunktangabe zurück. Wenn die Dimension der Quellengeometrie 0 ist (beispielsweise ein Punkt und eine Mehrpunktangabe), ist eine exakte Übereinstimmung erforderlich; für diese Punkte wird ein übereinstimmendes Maß als Mehrpunktangabe zurückgegeben. Für Quellengeometrien mit einer Dimension größer als Null wird die Position interpoliert. Wenn das Maß beispielsweise als 5.5 eingegeben wird und die Maße an den Scheitelpunkten einer Linienfolge 3, 4, 5, 6 und 7 sind, wird der interpolierte Punkt, der exakt zwischen den Scheitelpunkten 5 und 6 liegt, zurückgegeben.

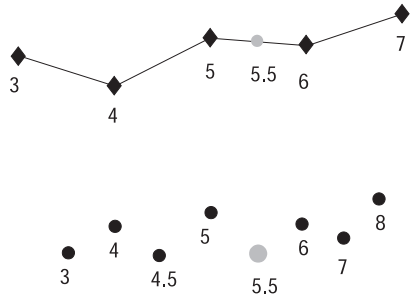


Abbildung 25. LocateAlong

Weitere Informationen hierzu finden Sie im Abschnitt „LocateAlong" auf Seite 179.

LocateBetween

Die Funktion LocateBetween gibt entweder die Gruppe der Pfade oder Positionen zurück zwischen zwei Maßen einer Quellengeometrie mit Maßen. Wenn die Dimension der Quellengeometrie 0 ist, gibt LocateBetween eine Mehrpunktangabe zurück mit allen Punkten, deren Maß zwischen den beiden Quellenmaßen liegt. Für Quellengeometrien mit einer Dimension größer als 0 gibt LocateBetween eine Mehrlinienfolge zurück, wenn ein Pfad interpoliert werden kann; andernfalls gibt LocateBetween eine Mehrpunktangabe mit den Punktpositionen zurück. Ein leerer Punkt wird zurückgegeben, wenn LocateBetween keinen Pfad interpolieren kann oder keine Position zwischen den Maßen gefunden wird. LocateBetween führt eine inklusive Suche der Geometrien durch; die Maße der Geometrien müssen daher größer oder gleich dem Maß *from* und kleiner oder gleich dem Maß *to* sein.

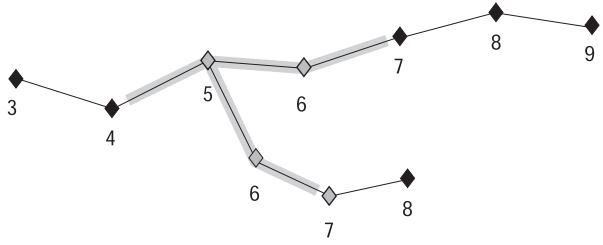


Abbildung 26. LocateBetween

Weitere Informationen hierzu finden Sie im Abschnitt „LocateBetween" auf Seite 181 .

ST_ConvexHull

Die Funktion ST_ConvexHull gibt ein konvexes Hülsenpolygon einer Geometrie zurück, die mindestens drei Scheitelpunkte haben muß, die eine konvexe Form bilden. Wenn die Scheitelpunkte der Geometrie keine konvexe Form bilden, gibt ST_ConvexHull Null zurück. ST_ConvexHull ist häufig der erste Schritt bei der Mosaikbildung zum Erstellen eines TIN-Netzwerks aus einer Reihe von Punkten.

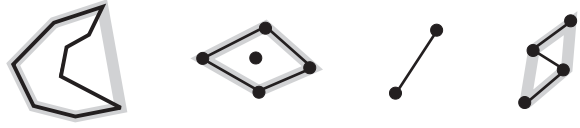


Abbildung 27. ST_ConvexHull

Weitere Informationen hierzu finden Sie im Abschnitt „ST_ConvexHull" auf Seite 204.

ST_Polygon

Generiert ein Polygon aus einer Linienfolge. Weitere Informationen hierzu finden Sie im Abschnitt „ST_Polygon" auf Seite 274.

Funktionen, die das Format der Werte einer Geometrie umsetzen

Der DB2 Spatial Extender unterstützt drei GIS-Datenaustauschformate:

- Bekannte Textdarstellung
- Bekannte Binärdarstellung
- ESRI binäre Formdarstellung

Bekannte Textdarstellung

Der DB2 Spatial Extender hat verschiedene Funktionen, die Geometrien aus Textbeschreibungen generieren.

ST_WKTToSQL
Erstellt eine Geometrie aus einer Textdarstellung eines beliebigen Geometrietyps. Es darf keine Kennung für ein räumliches Bezugssystem angegeben werden. Weitere Informationen hierzu finden Sie im Abschnitt „ST_WKTToSQL" auf Seite 287.

ST_GeomFromText
Erstellt eine Geometrie aus einer Textdarstellung eines beliebigen Geometrietyps. Es muß eine Kennung für ein räumliches Bezugssystem angegeben werden. Weitere Informationen hierzu finden Sie im Abschnitt „ST_GeometryFromText" auf Seite 222.

ST_PointFromText
Erstellt einen Punkt aus einer aus der Textdarstellung eines Punkts. Weitere Informationen hierzu finden Sie im Abschnitt „ST_PointFromText" auf Seite 265.

ST_LineFromText
Erstellt eine Linienfolge aus der Textdarstellung einer Linienfolge. Weitere Informationen hierzu finden Sie im Abschnitt „ST_LineFromText" auf Seite 248.

ST_PolyFromText
Erstellt ein Polygon aus der Textdarstellung eines Polygons. Weitere Informationen hierzu finden Sie im Abschnitt „ST_PolyFromText" auf Seite 271.

ST_MPointFromText
Erstellt eine Mehrpunktangabe aus der Textdarstellung einer Mehrpunktangabe. Weitere Informationen hierzu finden Sie im Abschnitt „ST_MPointFromText" auf Seite 254.

ST_MLineFromText
Erstellt eine Mehrlinienfolge aus der Darstellung einer Mehrlinienfolge. Weitere Informationen hierzu finden Sie im Abschnitt „ST_MLineFromText" auf Seite 251.

ST_MPolyFromText
Erstellt ein Multipolygon aus der Darstellung eines Multipolygons. Weitere Informationen hierzu finden Sie im Abschnitt „ST_MPolyFromText" auf Seite 256.

Die Textdarstellung ist eine ASCII-Zeichenfolge. Sie ermöglicht den Austausch der Geometrie in ASCII-Textform. Diese Funktionen erfordern keine Definition spezieller Programmstrukturen für die Zuordnung einer binären Darstellung. Diese Funktionen können daher in einem 3GL- oder 4GL-Programm verwendet werden.

Die Funktion ST_AsText setzt eine vorhandene Geometrie um in eine Textdarstellung. Weitere Informationen hierzu finden Sie im Abschnitt „ST_AsText" auf Seite 196.

Eine ausführliche Beschreibung der bekannten Textdarstellungen finden Sie im Abschnitt „Die bekannten OGC-Textdarstellungen" auf Seite 303.

Bekannte Binärdarstellung

Der DB2 Spatial Extender hat verschiedene Funktionen, die Geometrien aus bekannten binären Darstellungen (WKB-Darstellungen) generieren.

ST_WKBToSQL
Erstellt eine Geometrie aus einer bekannten Binärdarstellung eines beliebigen Geometrietyps. Es darf keine Kennung für ein räumliches Bezugssystem angegeben werden. Weitere Informationen hierzu finden Sie im Abschnitt „ST_WKBToSQL" auf Seite 285.

ST_GeomFromWKB
Erstellt eine Geometrie aus einer bekannten Binärdarstellung eines beliebigen Geometrietyps. Es muß eine Kennung für ein räumliches Bezugssystem angegeben werden. Weitere Informationen hierzu finden Sie im Abschnitt „ST_GeomFromWKB" auf Seite 224.

ST_PointFromWKB
Erstellt einen Punkt aus einer bekannten Binärdarstellung eines beliebigen Punkts. Weitere Informationen hierzu finden Sie im Abschnitt „ST_PointFromWKB" auf Seite 266.

ST_LineFromWKB
Erstellt eine Linienfolge aus einer bekannten Binärdarstellung einer Linienfolge. Weitere Informationen hierzu finden Sie im Abschnitt „ST_LineFromWKB" auf Seite 249.

ST_PolyFromWKB
Erstellt ein Polygon aus einer bekannten Binärdarstellung eines Polygons. Weitere Informationen hierzu finden Sie im Abschnitt „ST_PolyFromWKB" auf Seite 272.

ST_MPointFromWKB
Erstellt eine Mehrpunktangabe aus einer bekannten Binärdarstellung einer Mehrpunktangabe. Weitere Informationen hierzu finden Sie im Abschnitt „ST_MPointFromWKB" auf Seite 255.

ST_MLineFromWKB
Erstellt eine Mehrlinienfolge aus einer bekannten Binärdarstellung einer Mehrlinienfolge. Weitere Informationen hierzu finden Sie im Abschnitt „ST_MLineFromWKB" auf Seite 252.

ST_MPolyFromWKB
Erstellt ein Multipolygon aus einer bekannten Binärdarstellung eines Multipolygons. Weitere Informationen hierzu finden Sie im Abschnitt „ST_MPolyFromWKB" auf Seite 257.

Die bekannte Binärdarstellung ist ein fortlaufender Byte-Datenstrom. Sie ermöglicht den Austausch der Geometrie zwischen einem ODBC-Client und einer SQL-Datenbank in binärer Form. Diese Geometriefunktionen erfordern die Definition von C-Strukturen für die Zuordnung der binären Darstellung. Sie sind somit für die Verwendung in einem 3GL-Programm konzipiert und eignen sich nicht für 4GL-Umgebungen.

Die Funktion ST_AsBinary setzt eine vorhandene Geometrie um in eine Binärdarstellung. Weitere Informationen hierzu finden Sie im Abschnitt „ST_AsBinary" auf Seite 195.

Eine ausführliche Beschreibung der bekannten Binärdarstellungen finden Sie im Abschnitt „Die bekannten OGC-Binärdarstellungen (WKB)" auf Seite 308.

ESRI-Formdarstellung

Der DB2 Spatial Extender hat verschiedene Funktionen, die Geometrien aus ESRI-Formdarstellungen generieren. Die ESRI-Formdarstellung unterstützt zusätzlich zu den von Text- und Binärdarstellungen unterstützten zweidimensionalen Darstellungen auch Z-Koordinaten und Maße.

ShapeToSQL
Erstellt eine Geometrie aus einer Form eines beliebigen Geometrietyps. Es darf keine Kennung für ein räumliches Bezugssystem angegeben werden. Weitere Informationen hierzu finden Sie im Abschnitt „ShapeToSQL" auf Seite 191.

GeometryFromShape
Erstellt eine Geometrie aus einer Form eines beliebigen Geometrietyps. Es muß eine Kennung für ein räumliches Bezugssystem angegeben werden. Weitere Informationen hierzu finden Sie im Abschnitt „GeometryFromShape" auf Seite 171.

PointFromShape
Erstellt einen Punkt aus einer Punktform. Weitere Informationen hierzu finden Sie im Abschnitt „PointFromShape" auf Seite 188.

LineFromShape
Erstellt eine Linienfolge aus einer Mehrlinienform. Weitere Informationen hierzu finden Sie im Abschnitt „LineFromShape" auf Seite 177.

PolyFromShape
Erstellt ein Polygon aus einer Mehrlinienform. Weitere Informationen hierzu finden Sie im Abschnitt „PolyFromShape" auf Seite 189.

MPointFromShape
Erstellt eine Mehrpunktangabe aus der Form einer Mehrpunktangabe. Weitere Informationen hierzu finden Sie im Abschnitt „MPointFromShape" auf Seite 186.

MLineFromShape
Erstellt eine Mehrlinienfolge aus der Form einer Mehrlinienfolge. Weitere Informationen hierzu finden Sie im Abschnitt „MLine FromShape" auf Seite 184.

MPolyFromShape
Erstellt ein Multipolygon aus der Form eines Polygons aus mehreren Teilen. Weitere Informationen hierzu finden Sie im Abschnitt „MPolyFromShape" auf Seite 187.

Die allgemeine Syntax dieser Funktionen ist gleich. Das erste Argument ist die Formdarstellung, die als BLOB-Datentyp eingegeben wird. Das zweite Argument ist die Kennung des räumlichen Bezugs, die der Geometrie zugeordnet werden soll. Die Funktion GeometryFromShape hat beispielsweise die folgende Syntax:

```
GeometryFromShape(shapegeometry, SRID)
```

Zur Zuordnung der Binärdarstellung erfordern diese Formfunktionen die Definition einer C-Struktur. Sie sind somit für die Verwendung in einem 3GL-Programm konzipiert und eignen sich nicht für 4GL-Umgebungen.

Die Funktion AsBinaryShape setzt einen Geometriewert um in eine ESRI-Formdarstellung. Weitere Informationen hierzu finden Sie im Abschnitt „AsBinaryShape" auf Seite 170.

Eine ausführliche Beschreibung der Formdarstellungen finden Sie im Abschnitt „Die ESRI-Formdarstellungen" auf Seite 313.

Kapitel 14. Räumliche Funktionen für SQL-Abfragen

Dieses Kapitel enthält eine Liste der zur Abfrage räumlicher Daten verfügbaren Funktionen. Jede Funktion wird in einem Abschnitt mit Syntax, Rückgabetyp und Codebeispielen beschrieben. Einige der Beispiele in diesem Kapitel enthalten eine Anweisung CREATE TABLE, in der eine oder mehrere Spalten als räumliche Spalten definiert sind.

Folgende Überlegungen sind für räumliche Salten relevant:

- Die Beispiele in diesem Kapitel sind mit dem Bibliotheksnamen db2gse angegeben. Statt jede räumliche Funktion und jeden Typ mit db2gse explizit anzugeben, können Sie den Funktionspfad auch so einstellen, daß er db2gse enthält.
- Bevor Sie Daten in eine räumliche Spalte eingeben können:
  - Eventuell müssen Sie den Wert für den Parameter udf_mem_sz vergrößern. Der vorgeschlagene Anfangswert ist lautet 2048. Wenn 2048 nicht geeignet ist, vergrößern Sie den Parameter udf_mem_sz in Schritten zu 256.
  - Sie müssen diesen Parameter als Schicht registrieren. Weitere Informationen zum Registrieren einer räumlichen Spalte als Schicht finden Sie in „Kapitel 4. Räumliche Spalten definieren, als Schichten registrieren und Geocodierer aktivieren" auf Seite 35.

AsBinaryShape

AsBinaryShape verwendet ein Geometrieobjekt und gibt ein BLOB zurück.

Syntax

db2gse.AsBinaryShape(g db2gse.ST_Geometry)

Rückgabetyp

BLOB(1m)

Beispiele

Der folgende Codeausschnitt zeigt, wie die Funktion AsBinaryShape die Zonenpolygone der Tabelle SENSITIVE_AREAS in Formpolygone umwandelt. Diese Formpolygone werden an die Funktion draw_polygon zur Anzeige übergeben.

```
/* 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 verwendet eine Form und die Identität eines räumlichen Bezugssystems und gibt ein Geometrieobjekt zurück.

Syntax

db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Der folgende C-Codeausschnitt enthält ODBC-Funktionen, die in SQL-Funktionen des DB2 Spatial Extender eingebettet sind; diese Funktionen fügen Daten in die Tabelle LOTS ein.

Die Tabelle LOTS wurde mit zwei Spalten erstellt: der Spalte LOT_ID, die jedes Grundstück eindeutig kennzeichnet, und der Polygonspalte LOT, die die Geometrie jedes Grundstücks enthält.

```
CREATE TABLE LOTS ( lot_id  integer,
                    lot     db2gse.ST_MultiPolygon);
```

Die Funktion GeometryFromShape wandelt Formen in eine DB2 Spatial Extender-Geometrie um. Die gesamte Anweisung INSERT wird in shp_sql kopiert. Die Anweisung INSERT enthält Parametermarken zum dynamischen Akzeptieren der LOT_ID-Daten und der LOT-Daten.

```
/* 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 gibt 1 (TRUE) zurück, wenn sich die Umrisse von zwei Geometrien schneiden; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.EnvelopesIntersect(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die Funktion get_window ruft die Koordinaten des Anzeigefensters aus der Anwendung ab. Der Fensterparameter ist tatsächlich eine Struktur in Form eines Polygons mit einer Zeichenfolge von Koordinaten, die das Anzeigepolygon darstellen. Die Funktion PolyFromShape wandelt die Form des Anzeigefensters in ein DB2 Spatial Extender-Polygon um, das die Funktion EnvelopesIntersect als ihren Schnittumriß verwendet. Alle Zonenpolygone SENSITIVE_AREAS, die den inneren Bereich oder die Begrenzung des Anzeigefensters schneiden, werden zurückgegeben. Jedes Polygon wird aus dem Ergebnis abgerufen und an die Funktion draw_polygon weitergegeben.

```
/* 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 verwendet ein Geometrieobjekt und gibt 1 (TRUE) zurück, wenn das Objekt 3D-Koordinaten hat; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.Is3d(g db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle THREED_TEST, die zwei Spalten enthält: die Spalte GID des Typs Integer und die Spalte G1.

```
CREATE TABLE THREED_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Die Anweisung INSERT fügt zwei Punkte in die Tabelle THREED_TEST ein. Der erste Punkt enthält keine Z-Koordinaten, der zweite dagegen schon.

```
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)))
```

Die folgende Anweisung SELECT listet den Inhalt der Spalte GID mit den Ergebnissen der Funktion Is3d auf. Die Funktion gibt 0 für die erste Zeile zurück; dieser Wert enthält keine Z-Koordinaten. Für die zweite Zeile wird 1 zurückgegeben; dieser Wert enthält Z-Koordinaten.

```
SELECT gid, db2gse.Is3d (g1) "Ist es 3d?" FROM THREED_TEST
```

Das folgende Ergebnis wird zurückgegeben.

```
gid      Ist es 3d?
------   ----------
     1            0
     2            1
```

IsMeasured

IsMeasured verwendet ein Geometrieobjekt und gibt 1 (TRUE) zurück, wenn das Objekt Maße hat; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.IsMeasured(g db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle MEASURE_TEST mit zwei Spalten. Die GID-Spalte kennzeichnet die Zeilen eindeutig, und die Spalte G1 speichert die Punktgeometrien.

```
CREATE TABLE MEASURE_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Die folgenden INSERT-Anweisungen fügen zwei Datensätze in die Tabelle MEASURE_TEST ein. Der erste Datensatz speichert einen Punkt, der kein Maß hat. Der Punkt des zweiten Datensatzes hat ein Maß.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe zeigen die GID-Spalte zusammen mit den Ergebnissen der Funktion IsMeasured. Die Funktion IsMeasured gibt 0 für die erste Zeile zurück, da der Punkt kein Maß hat. Sie gibt 1 für die zweite Zeile zurück, da der Punkt Maßangaben hat.

```
SELECT gid, db2gse.IsMeasured (g1) "Has measures?" FROM MEASURE_TEST
gid      Has measures
------   ----------
     1            0
     2            1
```

LineFromShape

LineFromShape verwendet eine Form des Typs Point (Punkt) und die Identität eines räumlichen Bezugssystems und gibt eine Linienfolge (Linestring) zurück.

Syntax

db2gse.Line FromShape(ShapeLineString Blob(1M), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_LineString

Beispiele

Der folgende Codeabschnitt füllt die Tabelle SEWERLINES mit der eindeutigen ID, der Größenklasse und der Geometrie aller Abwasserkanallinien auf.

Die Anweisung CREATE TABLE erstellt die Tabelle SEWERLINES mit drei Spalten. Die erste Spalte, SEWER_ID, kennzeichnet jede Abwasserkanallinie eindeutig. Die zweite Spalte CLASS des Typs Integer kennzeichnet den Typ des Abwasserkanals; normalerweise gibt diese Spalte die Kapazität des Kanals an. Die dritte Spalte, SEWER, des Typs Linienfolge speichert die Geometrie der Abwasserkanallinie.

```
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 verwendet ein Geometrieobjekt und ein Maß und gibt die in diesem Maß gefundenen Punkte als Mehrpunktergebnis (Multipoint) zurück.

Syntax

db2gse.LocateAlong(g db2gse.ST_Geometry, adistance Double)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle LOCATEALONG_TEST. LOCATEALONG_TEST enthält zwei Spalten: die Spalte GID, die jede Zeile eindeutig kennzeichnet, und die Spalte G1, in der die Mustergeometrie gespeichert wird.

```
CREATE TABLE LOCATEALONG_TEST (gid integer, g1 db2gse.ST_Geometry)
```

Die folgenden INSERT-Anweisungen fügen zwei Zeilen ein. Die erste ist eine Mehrlinienfolge (Multilinestring), die zweite eine Mehrpunktangabe (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)))
```

In der folgenden Anweisung SELECT und der entsprechenden Ergebnisgruppe wird die Funktion LocateAlong angewiesen, die Punkte mit dem Maß 6.5 zu finden. Die erste Zeile gibt eine Mehrpunktangabe (Multipoint) mit zwei Punkten zurück. Die zweite Zeile gibt dagegen einen leeren Punkt zurück. Für lineare Funktionen (Geometrie mit einer Abmessung größer als 0) kann LocateAlong den Punkt interpolieren; bei Mehrpunktangaben (Multipoint) muß das Zielmaß jedoch exakt übereinstimmen.

```
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.
```

In der folgenden Anweisung SELECT und der entsprechenden Ergebnisgruppe gibt die Funktion LocateAlong für beide Punkte Mehrpunktangaben (Multipoint) zurück. Das Zielmaß von 7 entspricht den Maßen in den Mehrlinien- und Mehrpunkt-Quellendaten.

```
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 verwendet ein Geometrieobjekt und zwei Meßstandorte und gibt eine Geometrie zurück, die die Gruppe der unterbrochenen Pfade zwischen den beiden Meßstandorten darstellt.

Syntax

db2gse.LocateBetween(g db2gse.ST_Geometry, adistance Double, anotherdistance Double)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle LOCATEBETWEEN_TEST. LOCATEBETWEEN_TEST enthält zwei Spalten: die Spalte GID, die jede Zeile eindeutig kennzeichnet, und die Mehrlinienspalte G1, in der die Mustergeometrie gespeichert wird.

```
CREATE TABLE LOCATEBETWEEN_TEST (gid integer, g1 db2gse.ST_Geometry)
```

Die folgenden INSERT-Anweisungen fügen zwei Zeilen in die Tabelle LOCATEBETWEEN_TEST ein. Die erste Zeile ist eine Mehrlinienfolge (Multilinestring) , die zweite eine Mehrpunktangabe (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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe zeigen, wie die Funktion LocateBetween Maße zwischen den Angaben 6.5 und 7.5 (je einschließlich) findet. Die erste Zeile gibt eine Mehrlinienfolge mit mehreren Linienfolgen (Linestrings) zurück. Die zweite Zeile gibt eine Mehrpunktangabe (Multipoint) zurück, da es sich bei den Quellendaten um Mehrpunktangaben handelte. Wenn die Quellendaten eine Dimension von 0 (Punkt oder Mehrpunkt) haben, ist eine exakte Übereinstimmung erforderlich.

```
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 verwendet einen Punkt und gibt sein Maß zurück.

Syntax

db2gse.M(p db2gse.ST_Point)

Rückgabetyp

Double

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle M_TEST. M_TEST enthält zwei Spalten: die Integer-Spalte GID, die die Zeile eindeutig kennzeichnet, und die Point-Spalte PT1, in der die Mustergeometrie gespeichert wird.

```
CREATE TABLE M_TEST (gid integer, pt1 db2gse.ST_Point)
```

Die folgenden INSERT-Anweisungen fügen eine Zeile ein, die einen Punkt mit Maßangaben enthält, und eine Zeile mit einem Punkt ohne Maß.

```
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)))
```

In der folgenden Anweisung SELECT und der entsprechenden Ergebnisgruppe listet die Funktion M die Maßwerte der Punkte auf. Da der erste Punkt kein Maß hat, gibt die Funktion M den Wert NULL zurück.

```
SELECT gid, db2gse.M (pt1) "The measure" FROM M_TEST

GID         The measure
----------- ------------------------
          1                        -
          2   +7.00000000000000E+000

  2 record(s) selected.
```

MLine FromShape

MLineFromShape verwendet eine Form des Typs Mehrlinienfolge (Multilinestring) und die Identität eines räumlichen Bezugssystems und gibt eine Mehrlinienfolge zurück.

Syntax

db2gse.MLineFromShape(ShapeMultiLineString Blob(1M), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_MultiLineString

Beispiele

Der folgende Codeabschnitt füllt die Tabelle WATERWAYS mit einer eindeutigen ID, einem Namen und einer Mehrlinienfolge WATER aus.

Die Tabelle WATERWAYS wird erstellt mit den Spalten ID und NAME, die jeden in der Tabelle gespeicherten Fluß und Wasserlauf kennzeichnen. Die Spalte WATER ist eine Mehrlinienfolge, da der Fluß und die Flußsysteme häufig eine Zusammenfassung verschiedener Linienfolgen sind.

```
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 verwendet eine Form des Typs Multipoint (Mehrpunkt) und die Identität eines räumlichen Bezugssystems und gibt eine Mehrpunktangabe zurück.

Syntax

db2gse.MPointFromShape(ShapeMultiPoint (1M), srs db2gse.coordref)

Rückgabetyp

db2gse.ST_MultiPoint

Beispiele

Dieser Codeabschnitt füllt die Tabelle SPECIES_SITINGS eines Biologen aus.

Die Tabelle SPECIES_SITINGS wird mit drei Spalten erstellt. Die Spalten "species" und "genus" kennzeichnen jede Zeile eindeutig, während die Mehrpunktangabe "sitings" die Standorte der Spezies speichert.

```
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 verwendet eine Form des Typs Multipolygon und die Identität eines räumlichen Bezugssystems und gibt ein Multipolygon zurück.

Syntax

db2gse.MPolyFromShape(ShapeMultiPolygon Blob(1m), srs db2gse.coordref)

Rückgabetyp

db2gse.ST_MultiPolygon

Beispiele

Dieser Codeabschnitt füllt die Tabelle LOTS aus.

Die Tabelle LOTS speichert die lot_id, die jedes Grundstück eindeutig kennzeichnet und das Multipolygon für das Grundstück, die die Geometrie für die Grundstückslinie enthält.

```
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 verwendet eine Form des Typs Point (Punkt) und die Identität eines räumlichen Bezugssystems und gibt einen Punkt zurück.

Syntax

db2gse.PointFromShape(db2gse.ShapePoint blob(1M), srs db2gse.coordref)

Rückgabetyp

db2gse.ST_Point

Beispiele

Dieser Programmabschnitt füllt die Tabelle HAZARDOUS_SITES aus.

Die Gefahrenstandorte werden in der Tabelle HAZARDOUS_SITES gespeichert, die mit der nachfolgenden Anweisung CREATE TABLE erstellt wird. Die als Punkt definierte Spalte location speichert einen Standort, der das geographische Zentrum jedes Gefahrenstandorts darstellt.

```
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 verwendet eine Form des Typs Polygon und die Identität eines räumlichen Bezugssystems und gibt ein Polygon zurück.

Syntax

db2gse.PolyFromShape (ShapePolygon Blob(1M), srs db2gse.coordref)

Rückgabetyp

db2gse.ST_Polygon

Beispiele

Dieser Programmabschnitt füllt die Tabelle SENSITIVE_AREAS aus. Die Fragezeichen sind Parametermarken für die Werte ID, den Namen, die Größe, den Typ und die Zone, die zur Laufzeit abgerufen werden.

Die Tabelle SENSITIVE_AREAS enthält verschiedene Spalten, die die gefährdeten Institutionen beschreiben, zusätzlich zu der Spalte zone, die die Polygongeometrie der Institution beschreibt.

```
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 konstruiert einen Wert für db2gse.ST_Geometry mit seiner bekannten binären Darstellung. Der SRID-Wert Null wird automatisch verwendet.

Syntax

db2gse.ST_ShapeToSQL(ShapeGeometry blob(1M))

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Der folgende C-Codeausschnitt enthält ODBC-Funktionen, die in SQL-Funktionen des DB2 Spatial Extender eingebettet sind; diese Funktionen fügen Daten in die Tabelle LOTS ein. Die Tabelle LOTS wurde mit zwei Spalten erstellt: der Spalte lot_id, die jedes Grundstück eindeutig kennzeichnet, und der Multipolygonspalte lot, die die Geometrie jedes Grundstücks enthält.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

Die Funktion ShapeToSQL wandelt Formen in eine DB2 Spatial Extender-Geometrie um. Die gesamte Anweisung INSERT wird in shp_sql kopiert. Die Anweisung INSERT enthält Parametermarken zum dynamischen Akzeptieren der LOT_ID-Daten und der LOT-Daten.

```
/* 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 verwendet ein Polygon oder ein Multipolygon und gibt seine Fläche zurück.

Syntax

db2gse.ST_Area(s db2gse.ST_Surface)

Rückgabetyp

Double

Beispiele

Der Stadtplaner benötigt eine Liste der Gebäude. Zum Abrufen der Liste wählt ein GIS-Techniker die Gebäude-ID und die Grundfläche der einzelnen Gebäude aus.

Die Gebäudegrundflächen sind in der Tabelle BUILDINGFOOTPRINTS gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wird:

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

Für die Anforderung des Stadtplaners verwendet der Techniker die folgende Anweisung SELECT zur Auswahl des eindeutigen Schlüssels, der building_id und der Grundfläche jedes Gebäudes aus der Tabelle BUILDINGFOOT-PRINTS:

```
SELECT building_id, db2gse.ST_Area (footprint) "Area"
FROM BUILDINGFOOTPRINTS;
```

Die Anweisung SELECT gibt die folgende Ergebnisgruppe zurück:

```
building_id   Area
------------  ------------------------
         506    +1.40768000000000E+003
        1208    +2.55759000000000E+003
         543    +1.80786000000000E+003
         178    +2.08671000000000E+003
         .
         .
         .
```

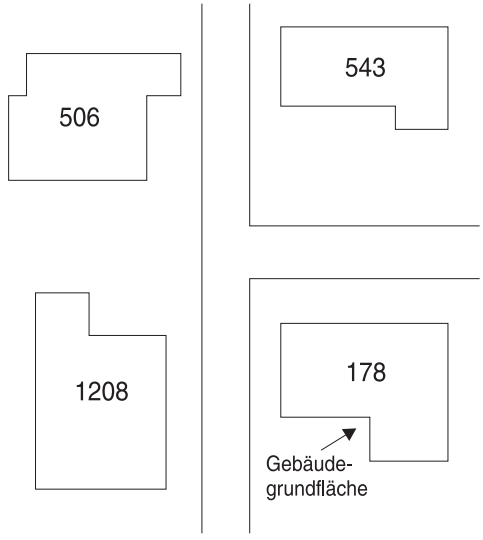


Abbildung 28. Über die Fläche eine Gebäudegrundfläche finden. Vier der fünf Gebäudegrundflächen mit ihren Gebäude-ID-Nummern werden entlang der angrenzenden Straße angezeigt.

ST_AsBinary

ST_AsBinary verwendet ein Geometrieobjekt und gibt seine bekannte binäre Darstellung zurück.

Syntax

db2gse.ST_AsBinary(g db2gse.ST_Geometry)

Rückgabetyp

BLOB(1m)

Beispiele

Der folgende Codeabschnitt verdeutlicht, wie die Funktion ST_AsBinary die Flächen-Multipolygone der Tabelle BUILDINGFOOTPRINTS in WKB-Multipolygone umwandelt. Diese Multipolygone werden an die Funktion draw_polygon zur Anzeige übergeben.

```
/* 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 verwendet ein Geometrieobjekt und gibt seine bekannte Textdarstellung zurück.

Syntax

db2gse.ST_AsText(g db2gse.ST_Geometry)

Rückgabetyp

Varchar(4000)

Beispiele

Im folgenden Szenario wandelt die Funktion db2gse.ST_AsText den Punkt HAZARDOUS_SITES ″location″ in seine Textbeschreibung um:

```
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;
```

Die Anweisung SELECT gibt die folgende Ergebnisgruppe zurück:

```
SITE_ID Name                                Location
------- ----------------------------------- ---------------------------------
    102 W. H. Kleenare Chemical Repository  POINT (1020.00000000 324.00000000)
```

ST_Boundary

ST_Boundary verwendet ein Geometrieobjekt und gibt seine kombinierte Begrenzung als Geometrieobjekt zurück.

Syntax

db2gse.ST_Boundary(g db2gse.ST_Geometry)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Im folgenden Codeabschnitt wird eine Tabelle mit dem Namen BOUNDARY_TEST erstellt. BOUNDARY_TEST enthält zwei Spalten: GEOTYPE, als Varchar definiert, und G1, als die Superklassengeometrie definiert. Die folgenden Anweisungen INSERT fügen jeweils eine der Unterklassengeometrien ein. Die Funktion ST_Boundary ruft die Begrenzungen jeder Unterklasse ab, die in der Geometriespalte G1 gespeichert ist. Beachten Sie, daß die Dimension der resultierenden Geometrie immer um eins niedriger ist als die Eingabegeometrie. Punkte und Mehrpunktangaben führen immer zu einer Begrenzung in einer leeren Geometrie, Dimension 1. Linienfolgen und Mehrlinienfolgen geben eine Mehrpunktbegrenzung, Dimension 0, zurück. Ein Polygon oder ein Multipolygon gibt immer eine Mehrlinienbegrenzung, Dimension 1, zurück.

```
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 verwendet ein Geometrieobjekt und eine Entfernung und gibt das Geometrieobjekt zurück, das das Quellenobjekt umgibt.

Syntax

db2gse.ST_Buffer(g db2gse.ST_Geometry , adistance Double)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Der zuständige Planer benötigt eine Liste der Gefahrenstandorte, in deren Umkreis von fünf Meilen sensible Bereiche wie Schulen, Krankenhäuser und Pflegeheime liegen. Die sensiblen Bereiche sind in der Tabelle SENSITIVE_AREAS gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wird. Die Spalte ZONE ist als Polygon definiert; dieses Polygon stellt den Umriß der jeweiligen sensiblen Bereiche dar.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

Die Gefahrenstandorte sind in der Tabelle HAZARDOUS_SITES gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wird. Die als Punkt definierte Spalte LOCATION speichert einen Standort, der das geographische Zentrum jedes Gefahrenstandorts darstellt.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

Die Tabellen SENSITIVE_AREAS und HAZARDOUS_SITES werden über die Funktion db2gse.ST_Overlaps verknüpft. Die Funktion gibt 1 (TRUE) zurück für alle Zeilen SENSITIVE_AREAS, deren Zonenpolygone den Puffer von fünf Meilen um den Punkt HAZARDOUS_SITES schneiden.

```
SELECT sa.name "Sensible Bereiche", hs.name "Gefahrenpunkte"
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Overlaps(sa.zone, db2gse.ST_Buffer (hs.location,(5 * 5280)))
=1;
```

In Abb. 29 auf Seite 200 liegen einige der sensiblen Bereiche in diesem Verwaltungsbezirk innerhalb des Fünf-Meilen-Puffers um die Gefahrenstandorte. Beide Fünf-Meilen-Puffer enthalten das Krankenhaus, einer davon enthält die Schule. Das Pflegeheim liegt außerhalb beider Bereiche.

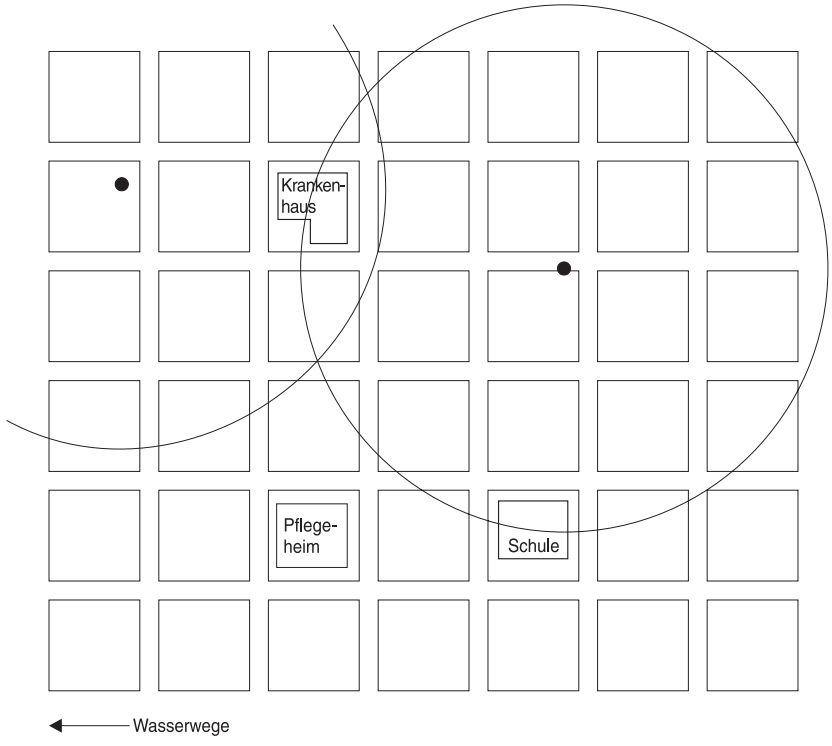


Abbildung 29. Ein Puffer mit einem Fünf-Meilen-Radius wird auf einen Punkt angewendet

ST_Centroid

ST_Centroid verwendet ein Polygon oder ein Multipolygon und gibt seinen geometrischen Mittelpunkt als Punkt zurück.

Syntax

db2gse.ST_Centroid(s db2gse.ST_Surface)
db2gse.ST_Centroid(ms db2gse.ST_MultiSurface)

Rückgabetyp

Für die Oberfläche: db2gse.ST_Point

Beispiele

Der städtische GIS-Techniker möchte die Multipolygone der Gebäudegrundflächen als einzelne Punkte in einer Grafik zur Bebauungsdichte anzeigen.

Die Gebäudegrundflächen sind in der Tabelle BUILDINGFOOTPRINTS gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wird.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                                 footprint   db2gse.ST_MultiPolygon);
```

Die Funktion ST_Centroid gibt den Mittelpunkt jedes Multipolygons für die Gebäudegrundflächen zurück. Die Funktion AsBinaryShape wandelt den Mittelpunkt in eine Form um, die von der Anwendung erkannte externe Darstellung.

```
SELECT building_id,
       CAST(db2gse.AsBinaryShape(db2gse.ST_Centroid (footprint)) as blob(1m))
"Centroid"
FROM BUILDINGFOOTPRINTS;
```

ST_Contains

ST_Contains verwendet zwei Geometrieobjekte und gibt 1 (TRUE) zurück, wenn das erste Objekt das zweite vollständig beinhaltet, andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_Contains(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Im nachfolgenden Beispiel werden zwei Tabellen erstellt. Eine Tabelle enthält die Gebäudegrundflächen (″footprints″) der Stadt, die andere Tabelle die Grundstücke (″lots″). Der Stadtplaner möchte sicherstellen, daß alle Gebäudegrundflächen vollständig auf dem jeweiligen Grundstück liegen.

In beiden Tabellen speichert der Datentyp Multipolygon die Geometrie der Gebäudegrundflächen und der Grundstücke. Der Datenbank-Designer hat für beide Funktionen Multipolygone ausgewählt. Der Designer hat festgestellt, daß die Grundstücke durch natürliche Merkmale wie beispielsweise Flüsse unterbrochen sein können und daß die Gebäudegrundflächen häufig mehrere Gebäude umfassen.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                                 footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

Der Stadtplaner wählt zunächst die Gebäude aus, die sich nicht vollständig auf einem Grundstück befinden.

```
SELECT building_id
FROM BUILDINGFOOTPRINTS, LOTS
WHERE db2gse.ST_Contains(lot,footprint) = 0;
```

Der Stadtplaner stellt fest, daß die erste Abfrage eine Liste aller Gebäude-IDs zurückgibt, deren Gebäudegrundfläche außerhalb eines Grundstückspolygons liegt. Er weiß aber auch, daß diese Information kein Hinweis darauf ist, ob den anderen Gebäuden die richtige Grundstücks-ID zugeordnet ist. Diese zweite Abfrage führt eine Datenintegritätsprüfung mit der Spalte lot_id der Tabelle BUILDINGFOOTPRINTS aus.

```
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;
```

In Abb. 30 liegen die mit ihren Gebäude-IDs gekennzeichneten Gebäudegrundflächen innerhalb der jeweiligen Grundstücke. Die Grundstückslinien sind als gepunktete Linien dargestellt. Dies ist zwar in der Abbildung nicht dargestellt; diese Linien reichen jedoch bis zur Mitte der Straße und umfassen die Grundstücke und die darauf enthaltenen Gebäudegrundflächen vollständig.

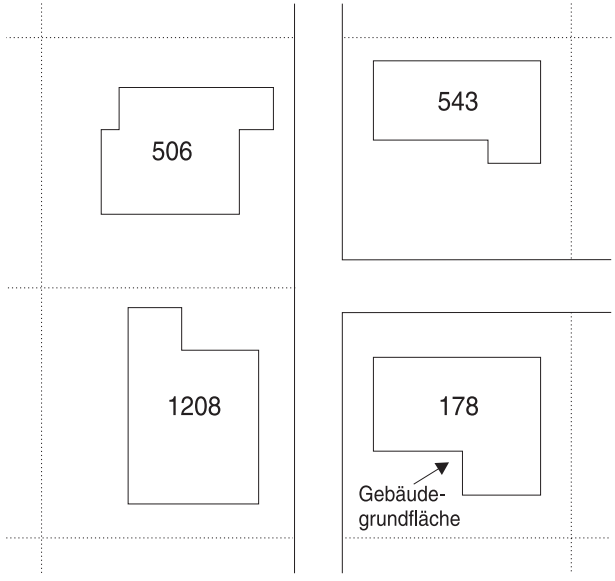


Abbildung 30. Mit ST_Contains sicherstellen, daß alle Gebäude vollständig auf dem jeweiligen Grundstück liegen

ST_ConvexHull

ST_ConvexHull verwendet ein Geometrieobjekt und gibt die konvexe Hülse zurück.

Syntax

db2gse.ST_ConvexHull(g db2gse.ST_Geometry)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Dieses Beispiel erstellt die Tabelle CONVEXHULL_TEST mit zwei Spalten: GEOTYPE und G1. Die Spalte GEOTYPE, eine Angabe varchar(20), speichert den Namen der Unterklasse einer in G1 gespeicherten Geometrie, die als Geometrie definiert ist.

```
CREATE TABLE CONVEXHULL_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Jede Anweisung INSERT fügt eine Geometrie jedes Unterklassentyps in die Tabelle CONVEXHULL_TEST ein.

```
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)))
```

Die folgende Anweisung SELECT listet den Unterklassennamen auf, der in der Spalte GEOTYPE und der konvexen Hülse gespeichert ist. Die von der Funktion ST_ConvexHull generierte konvexe Hülse wird mit der Funktion ST_AsText in Text umgewandelt. Anschließend wird sie in eine Angabe varchar(256) umgesetzt, da die Standardausgabe von ST_AsText varchar(4000) ist.

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText(db2gse.ST_ConvexHull(G1)))
as varchar(256) "The convexhull"
FROM CONVEXHULL_TEST
```

ST_CoordDim

ST_CoordDim gibt die Koordinatendimensionen des Werts ST_Geometry zurück. Eine Erläuterung der Koordinatendimensionen finden Sie im Abschnitt „Punkte” auf Seite 136.

Syntax

db2gse.ST_CoordDim(g1 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die Tabelle coorddim_test wird mit den Spalten geotype und g1 erstellt. Die Spalte geotype speichert den Namen der in der Geometriespalte g1 gespeicherten Geometrieunterklasse.

```
CREATE TABLE coorddim_test (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Die INSERT-Anweisungen fügen eine Muster-Unterklasse in die Tabelle coorddim_test ein.

```
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))
)
```

Die Anweisung SELECT listet den Unterklassennamen auf, der in der Spalte geotype mit der Koordinatendimension dieses Geotyps gespeichert ist.

```
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 verwendet zwei Geometrieobjekte und gibt 1 (TRUE) zurück, wenn deren Schnittmenge in einem Geometrieobjekt resultiert, deren Dimension um eins niedriger ist als die maximale Dimension der Quellenobjekte. Das Schnittobjekt enthält Punkte, die in beiden Quellengeometrieobjekten liegen; es ist jedoch nicht mit einem der Quellenobjekte identisch. Andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_Crosses(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die Regierung des Landkreises denkt über eine neue Vorschrift nach, die festlegt, daß alle Lagerstätten für gefährliche Abfälle im Landkreis nicht näher als fünf Meilen an Flußläufen oder Bächen liegen dürfen. Der zuständige GIS-Manager verfügt über eine exakte Darstellung der Flüsse und Bäche, die als Mehrlinienfolgen in der Tabelle WATERWAYS gespeichert sind. Für die Lagerstätten von gefährlichen Abfällen hat er jedoch nur einen einzigen Standortpunkt.

```
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);
```

Es soll festgestellt werden, ob die Landkreisregierung über Einrichtungen informiert werden muß, die die vorgeschlagene Vorschrift nicht erfüllen. Der GIS-Manager muß hierzu einen Bereich um die Gefahrenpunkte definieren und feststellen, ob sich Flüsse oder Bäche mit dem Polygon schneiden. ST_Crosses vergleicht die Bereiche aus HAZARDOUS_SITES mit WATERWAYS. Auf diese Weise werden nur die Datensätze zurückgegeben, bei denen ein Wasserlauf über den vorgeschlagenen Radius um den Gefahrenpunkt verläuft.

```
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;
```

In Abb. 31 schneidet der Fünf-Meilen-Puffer um den Gefahrenpunkt das Wassernetz des Landkreises. Das Netz der Wasserläufe wurde als Mehrlinienfolge definiert. Die Ergebnisgruppe umfaßt somit alle Liniensegmente, die Teil der Segmente sind, die den Radius schneiden.

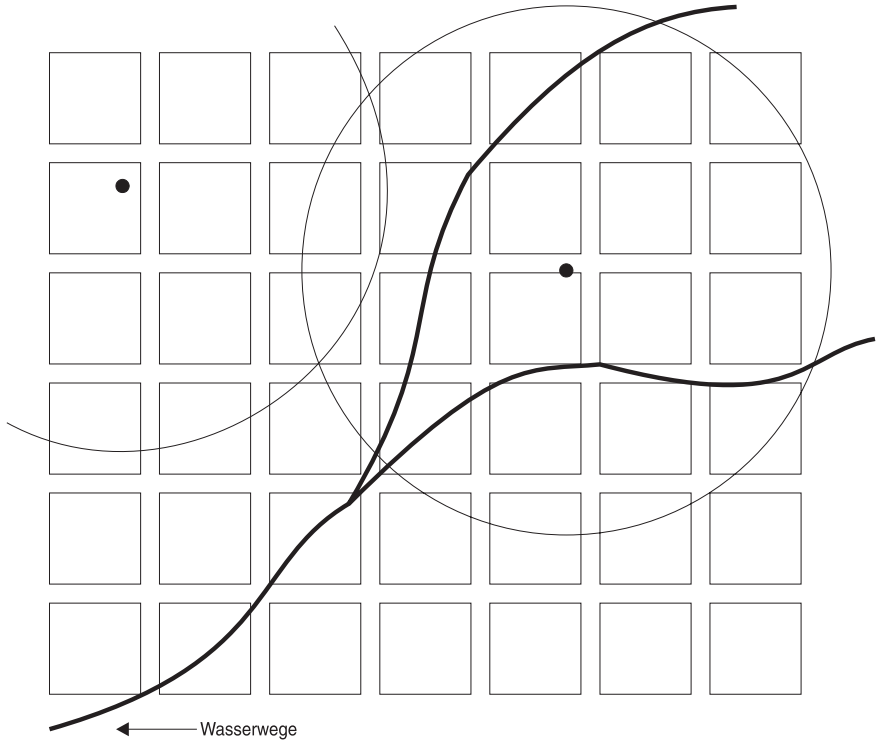


Abbildung 31. Mit ST_Crosses die Wasserwege ermitteln, die durch den Lagerungsbereich für gefährliche Abfälle fließen.

ST_Difference

ST_Difference verwendet zwei Geometrieobjekte und gibt ein Geometrieobjekt zurück, das die Differenz der beiden Eingabeobjekte darstellt.

Syntax

db2gse.ST_Difference(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Der Stadtplaner muß die Gesamtfläche der Grundstücke in der Stadt kennen, auf denen kein Gebäude steht. Das heißt, der Stadtplaner braucht die Summe der Fläche nach Abzug der bebauten Fläche.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                                 footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS ( lot_id  integer,
                    lot   db2gse.ST_MultiPolygon);
```

Der Stadtplaner erstellt eine Gleichsetzungsverknüpfung zwischen den Tabellen BUILDINGFOOTPRINTS und LOTS über die lot_id. Anschließend verwendet er die Flächensumme der Differenzen zwischen Grundstücke und den Gebäudegrundflächen.

```
SELECT SUM(db2gse.ST_Area(db2gse.ST_Difference(lot,footprint)))
FROM BUILDINGFOOTPRINTS bf, LOTS
WHERE bf.lot_id = LOTS.lot_id;
```

ST_Dimension

ST_Dimension verwendet ein Geometrieobjekt und gibt seine Dimension zurück.

Syntax

db2gse.ST_Dimension(g1 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die Tabelle DIMENSION_TEST wird mit den Spalten GEOTYPE und G1 erstellt. Die Spalte GEOTYPE speichert den Namen der in der Geometriespalte G1 gespeicherten Geometrieunterklasse.

```
CREATE TABLE DIMENSION_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Die INSERT-Anweisungen fügen eine Muster-Unterklasse in die Tabelle DIMENSION_TEST ein.

```
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)))
```

Die folgende Anweisung SELECT listet den Unterklassennamen auf, der in der Spalte GEOTYPE mit der Dimension dieses Geotyps gespeichert ist.

```
SELECT geotype, db2gse.ST_Dimension(g1) "The dimension"
FROM DIMENSION_TEST
```

Das folgende Ergebnis wird zurückgegeben.

```
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 verwendet zwei Geometrieobjekte und gibt 1 (TRUE) zurück, wenn die Schnittmenge der beiden Geometrien eine leere Menge ist; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_Disjoint(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Eine Versicherungsgesellschaft möchte die Höhe der Versicherungssumme für die Krankenhäuser, Pflegeheime und Schulen einer Stadt überprüfen. Im Rahmen dieses Prozesses wird auch die Gefährdung beurteilt, die die Lagerstätten für gefährliche Abfälle für die einzelnen Institutionen darstellen. Die Versicherungsgesellschaft möchte nur die Institutionen berücksichtigen, bei denen kein Risiko der Kontaminierung besteht. Der von der Versicherungsgesellschaft eingestellte GIS-Berater wurde angewiesen, alle Institutionen zu ermitteln, die nicht innerhalb eines Radius von fünf Meilen innerhalb einer Lagerstätte mit gefährlichen Abfällen liegen.

Die Tabelle SENSITIVE_AREAS enthält verschiedene Spalten, die die gefährdeten Institutionen beschreiben, zusätzlich zu der Spalte ZONE, die die Polygongeometrie der Institution beschreibt.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

In der Tabelle HAZARDOUS_SITES ist die Identität der Standorte in den Spalten SITE_ID und NAME gespeichert; der tatsächliche geographische Standort wird jeweils in der Spalte LOCATION gespeichert.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

Die folgende Anweisung SELECT listet die Namen aller sensiblen Bereiche auf, die sich nicht innerhalb des Fünf-Meilen-Radius von einer Lagerstätte gefährlicher Abfälle befinden. Die Funktion ST_Intersects könnte die Funktion ST_Disjoint in dieser Abfrage ersetzen, wenn das Ergebnis der Funktion auf 0 statt auf 1 gesetzt wird. ST_Intersects und ST_Disjoint geben genau entgegengesetzte Ergebnisse zurück.

```
SELECT sa.name
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Disjoint(db2gse.ST_Buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

In Abb. 32 werden sensible Standorte mit dem Fünf-Meilen-Radius um die Lagerstätten gefährlicher Abfälle verglichen. Das Pflegeheim ist der einzige sensible Bereich, für den die Funktion ST_Disjoint da Ergebnis 1 (TRUE) zurückgibt. Die Funktion ST_Disjoint gibt 1 zurück, wenn sich zwei Geometrien in keiner Weise schneiden.

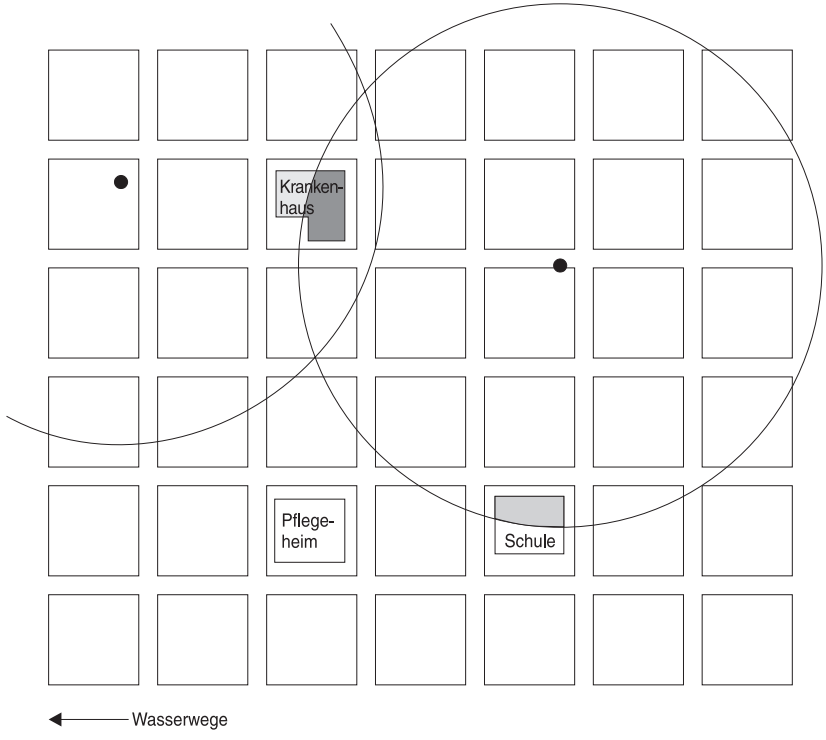


Abbildung 32. Mit ST_Disjoint die Gebäude ermitteln, die nicht innerhalb (in der Schnittmenge) eines Gefahrengebiets liegen

ST_Distance

ST_Distance verwendet zwei Geometrieobjekte und gibt die kleinste Entfernung zwischen diesen Objekten zurück.

Syntax

db2gse.ST_Distance(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Double

Beispiele

Der Stadtplaner benötigt eine Liste aller Gebäude, deren Entfernung bis zum Grundstücksrand maximal 1 Fuß (30,48 cm) beträgt.

Die Spalte BUILDING_ID der Tabelle BUILDINGFOOTPRINTS kennzeichnet jedes Gebäude eindeutig. Die Spalte LOT_ID kennzeichnet das Grundstück, zu dem das jeweilige Gebäude gehört. Das Multipolygon der Gebäudegrundfläche speichert die Geometrie der Grundfläche jedes Gebäudes.

```
CREATE TABLE BUILDINGFOOTPRINTS (  building_id integer,
                                   lot_id      integer,
                                   footprint   db2gse.ST_MultiPolygon);
```

Die Tabelle LOTS speichert die lot-ID, die jedes Grundstück eindeutig kennzeichnet sowie das Multipolygon, das die Geometrie der Grundstückslinie enthält.

```
CREATE TABLE LOTS ( lot_id  integer,
                        lot   db2gse.ST_MultiPolygon);
```

Die Abfrage gibt eine Liste der Gebäude-IDs zurück, die einen Abstand von maximal 1 Fuß zu der jeweiligen Grundstückslinie haben. Die Funktion ST_Distance führt eine räumliche Verknüpfung zwischen den Grundflächen und der Begrenzung der Grundstückspolygone durch. Die Gleichsetzungsverknüpfung zwischen bf.lot_id und LOTS.lot_id stellt jedoch sicher, daß nur die Multipolygone zu dem gleichen Grundstück mit der Funktion ST_Distance verglichen werden.

```
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 verwendet eine Linienfolge und gibt einen Punkt zurück, der den letzten Punkt der Linienfolge darstellt.

Syntax

db2gse.ST_Endpoint(c db2gse.ST_Curve)

Rückgabetyp

db2gse.ST_Point

Beispiele

Die Tabelle ENDPOINT_TEST speichert die ganzzahlige Spalte GID; die jede Zeile und die Linienspalte LN1, in der Mehrlinienfolgen gespeichert werden, eindeutig kennzeichnet.

```
CREATE TABLE ENDPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

Die Anweisungen INSERT fügt Linienfolgen in die Tabelle ENDPOINT_TEST ein. Die erste Zeile hat keine Z-Koordinaten oder Maßangaben; die zweite dagegen schon.

```
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)))
```

Die folgende Anweisung SELECT listet die Spalte GID mit den Ergebnissen der Funktion ST_Endpoint auf. Die Funktion ST_Endpoint generiert eine Punktgeometrie, die von der Funktion ST_AsText in Text umgewandelt wird. Mit der Funktion CAST wird der Standardwert varchar(4000) der Funktion ST_AsText auf varchar(60) verkürzt.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.ST_Endpoint(ln1)) AS varchar(60))
"Endpoint"
FROM ENDPOINT_TEST
```

Das folgende Ergebnis wird zurückgegeben.

```
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 verwendet ein Geometrieobjekt und gibt seinen Zeichenrahmen als Geometrie zurück.

Syntax

db2gse.ST_Envelope(g db2gse.ST_Geometry)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Die Spalte GEOTYPE in der Tabelle ENVELOPE_TEST speichert den Namen der in der Geometriespalte G1 gespeicherten Geometrieunterklasse.

```
CREATE TABLE ENVELOPE_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Die folgenden INSERT-Anweisungen fügen jede Geometrieunterklasse in die Tabelle ENVELOPE_TEST ein.

```
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)))
```

Die folgende Anweisung SELECT listet den Namen der Unterklasse neben ihrem Umschlag auf. Da die Funktion ST_Envelope entweder einen Punkt, eine Linienfolge oder ein Polygon zurückgibt, wird ihre Ausgabe mit der Funktion ST_AsText in Text umgewandelt. Die Funktion CAST wandelt das Standardergebnis varchar(4000) der Funktion ST_AsText in ein Ergebnis varchar(280) um.

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText(db2gse.ST_Envelope(g1)) AS varchar(280))
"The envelope"
FROM ENVELOPE_TEST
```

Das folgende Ergebnis wird zurückgegeben.

```
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 vergleicht zwei Geometrien und gibt 1 (TRUE) zurück, wenn die Geometrien identisch sind; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_Equals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Der GIS-Techniker vermutet, daß manche der Daten in der Tabelle BUILDINGFOOTPRINTS irgendwie dupliziert wurden. Der Techniker fragt die Tabelle ab, um festzustellen, ob irgendwelche der Multipolygone der Gebäudegrundflächen gleich sind.

Die Tabelle BUILDINGFOOTPRINTS wird mit der folgenden Anweisung erstellt. Die Spalte BUILDING_ID kennzeichnet die Gebäude eindeutig; die Spalte LOT_ID kennzeichnet das Grundstück zu dem Gebäude, und die Spalte FOOTPRINT speichert die Geometrie des Gebäudes.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

Die Tabelle BUILDINGFOOTPRINTS ist über das Prädikat ST_Equals mit sich selbst räumlich verknüpft; dieses Prädikat gibt 1 zurück, wenn es zwei gleiche Multipolygone findet. Die Bedingung bf1.building_id <> bf2.building_id ist erforderlich, um den Vergleich mit der gleichen Geometrie zu eliminieren.

```
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 verwendet ein Polygon und gibt seinen äußeren Ring als Linienfolge zurück.

Syntax

db2gse.ST_ExteriorRing(s db2gse.ST_Polygon)

Rückgabetyp

db2gse.ST_LineString

Beispiele

Ein Ornithologe, der den Vogelbestand auf verschiedenen Südseeinseln untersucht, weiß, daß sich die Zone für die Nahrungssuche einer bestimmten Spezies auf die Küstenlinien beschränkt. Zur Berechnung des Bestands, der auf der Insel genügend Nahrung finden könnte, muß er die Länge der Küstenlinie (den Umfang) kennen. Auf einigen der Inseln gibt es zwar mehrere Seen; deren Ufer werden jedoch ausschließlich von einer aggressiveren Spezies bevölkert. Der Ornithologe braucht daher nur die Länge des äußeren Inselrings zu kennen.

Die Spalten ID und NAME der Tabelle ISLANDS kennzeichnen alle Inseln, und die Spalte LAND des Typs ST_Polygon speichert die Geometrie dieser Inseln.

```
CREATE TABLE ISLANDS (id    integer,
                      name   varchar(32),
                      land   db2gse.ST_Polygon);
```

Die Funktion ST_ExteriorRing extrahiert den äußeren Ring von jedem Inselpolygon als Linienfolge. Die Länge der Linienfolge wird über die Funktion length ermittelt. Die Längen der Linienfolge werden über die Funktion SUM zusammengefaßt.

```
SELECT SUM(db2gse.ST_length(db2gse.ST_ExteriorRing (land))) FROM ISLANDS;
```

In Abb. 33 stellen die äußeren Ringe der Inseln die ökologische Schnittstelle zwischen den Inseln und dem Meer dar. Auf manchen der Inseln gibt es Seen, die als innere Ringe der Polygone dargestellt sind.

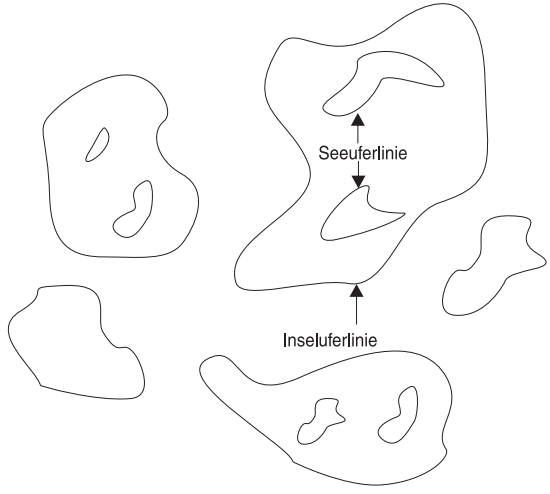


Abbildung 33. Mit ST_ExteriorRing die Länge der Küstenlinie einer Insel ermitteln.

ST_GeometryFromText

ST_GeometryFromText verwendet eine bekannte Textdarstellung und die Identität eines räumlichen Bezugssystems und gibt ein Geometrieobjekt zurück.

Syntax

db2gse.ST_GeometryFromText(geometryTaggedText Varchar(4000), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Die Tabelle GEOMETRY_TEST enthält die ganzzahlige Spalte GID, die jede Zeile eindeutig kennzeichnet, und die Spalte G1, in der die Geometrie gespeichert wird.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Die INSERT-Anweisungen fügen die Daten in die Spalten GID und G1 der Tabelle GEOMETRY_TEST ein. Die Funktion ST_GeometryFromText wandelt die Textdarstellung jeder Geometrie in die entsprechende DB2 Spatial Extender-Exemplarunterklasse um.

```
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 verwendet eine bekannte binäre Darstellung und die Identität eines räumlichen Bezugssystems und gibt ein Geometrieobjekt zurück.

Syntax

db2gse.ST_GeomFromWKB(WKBGeometry Blob(1M), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Der folgende C-Codeausschnitt enthält ODBC-Funktionen, die in SQL-Funktionen des DB2 Spatial Extender eingebettet sind; diese Funktionen fügen Daten in die Tabelle LOTS ein.

Die Tabelle LOTS wurde mit zwei Spalten erstellt: der Spalte LOT_ID, die jedes Grundstück eindeutig kennzeichnet, und der Multipolygonspalte LOT, die die Geometrie jedes Grundstücks enthält.

```
CREATE TABLE LOTS ( lot_id  integer,
                        lot   db2gse.ST_MultiPolygon);
```

Die Funktion ST_GeomFromWKB wandelt WKB-Darstellungen in DB2 Spatial Extender-Geometrien um. Die gesamte Anweisung INSERT wird in die Zeichenfolge wkb_sql kopiert. Die Anweisung INSERT enthält Parametermarken zum dynamischen Akzeptieren der LOT_ID-Daten und der LOT-Daten.

```
/* 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 verwendet eine Gruppe und einen ganzzahligen Index und gibt das *n*te Geometrieobjekt in der Gruppe zurück.

Syntax

db2gse.ST_GeometryN(g db2gse.ST_GeomCollection, n Integer)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Der Stadtplaner muß wissen, ob alle Gebäudegrundflächen innerhalb des ersten Polygons des Multipolygons des Grundstücks liegen.

Die Spalte BUILDING_ID kennzeichnet jede Zeile der Tabelle BUILDING-FOOTPRINTS eindeutig. Die Spalte LOT_ID kennzeichnet das Grundstück, zu dem die einzelnen Gebäude gehören. Die Spalte FOOTPRINT speichert die Geometrie des Gebäudes.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                         footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS ( lot_id  integer,
                    lot   db2gse.ST_MultiPolygon);
```

Die Abfrage listet die BUILDINGFOOTPRINTS building_id und die lot_id für alle Gebäudegrundflächen auf, die innerhalb des ersten Grundstückspolygons liegen. Die Funktion ST_GeometryN gibt das erste Grundstückspolygon in der Anordnung der Multipolygone zurück.

```
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 verwendet ein ST_Geometry-Objekt und gibt seinen Geometrietyp als Zeichenfolge zurück.

Syntax

db2gse.ST_GeometryType (g db2gse.ST_Geometry)

Rückgabetyp

Varchar(4000)

Beispiele

Die Tabelle GEOMETRYTYPE_TEST enthält die Geometriespalte G1.

```
CREATE TABLE GEOMETRYTYPE_TEST(g1 db2gse.ST_Geometry)
```

Die folgenden INSERT-Anweisungen fügen jede Geometrieunterklasse in die Spalte G1 ein.

```
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)))
```

Die folgende Anweisung SELECT listet den Geometrietyp jeder Unterklasse auf, die in der Geometriespalte G1 gespeichert ist.

```
SELECT db2gse.ST_GeometryType(g1) "Geometry type" FROM GEOMETRYTYPE_TEST
```

Das folgende Ergebnis wird zurückgegeben.

```
Geometry type
------------------------------
ST_Point
ST_LineString
ST_Polygon
ST_MultiPoint
ST_MultiLineString
ST_MultiPolygon

6 record(s) selected.
```

ST_InteriorRingN

Gibt den *n*en inneren Ring eines Polygons als Linienfolge zurück. Die Ringe sind nicht nach ihrer geometrischen Anordnung geordnet. Sie sind entsprechend den Regeln der internen geometrischen Prüfroutinen angeordnet. Die Reihenfolge der Ringe kann daher nicht vordefiniert werden.

Syntax

ST_InteriorRingN(p ST_Polygon, n Integer)

Rückgabetyp

db2gse.ST_LineString

Beispiele

Ein Ornithologe, der den Vogelbestand auf verschiedenen Südseeinseln untersucht, weiß, daß sich die Zone für die Nahrungssuche einer bestimmten passiven Spezies auf die Küstenlinie beschränkt. Auf manchen der Inseln gibt es kleine Seen. Die Ufer der Seen werden ausschließlich von einer aggressiveren Spezies bevölkert. Der Ornithologe weiß, daß ab einer bestimmten Uferlänge an den Seen die aggressive Spezies so zahlreich wird, daß sie die passive Spezies an der Küste bedroht. Der Ornithologe will daher die Summe der Uferlängen der inneren Ringe der Inseln wissen.

In Abb. 34 stellen die äußeren Ringe der Inseln die ökologische Schnittstelle zwischen den Inseln und dem Meer dar. Auf manchen der Inseln gibt es Seen, die als innere Ringe der Polygone dargestellt sind.

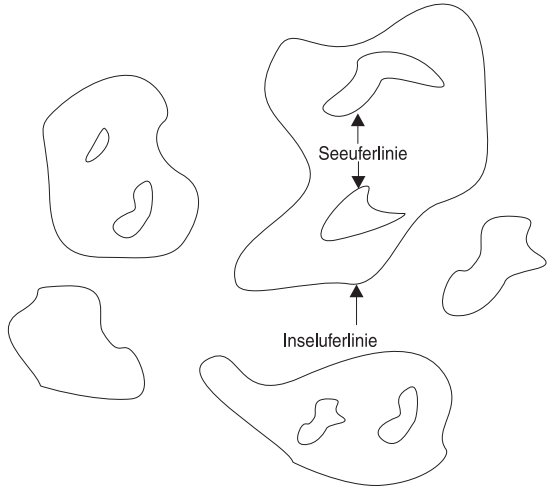


Abbildung 34. Mit ST_InteriorRingN die Länge der Seeufer auf allen Inseln ermitteln

Die Spalten ID und name der Tabelle ISLANDS kennzeichnen alle Inseln, und die Polygonspalte land speichert die Geometrie dieser Inseln.

```
CREATE TABLE ISLANDS (id    integer,
                      name  varchar(32),
                      land  db2gse.ST_Polygon);
```

Das folgende ODBC-Programm extrahiert mit der Funktion ST_InteriorRingN den inneren Ring (lake) aus jedem Inselpolygon als Linienfolge. Der Umfang der zurückgegebenen Linienfolge wird akkumuliert und zusammen mit der ID der Insel angezeigt.

```
#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))) an die Variable lake_perimeter binden */

  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 fertig ...\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 verwendet zwei Geometrieobjekte und gibt die Schnittmenge als Geometrieobjekt zurück.

Syntax

db2gse.ST_Intersection(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Der Chef der Feuerwehr muß die Bereiche der Krankenhäuser, Schulen, und Pflegeheime abrufen, die sich mit dem Radius einer möglichen Kontamination durch gefährliche Abfälle schneiden.

Die sensiblen Bereiche sind in der Tabelle SENSITIVE_AREAS gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wird. Die Spalte ZONE ist als Polygon definiert, in dem der Umriß der jeweiligen sensiblen Bereiche gespeichert ist.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

Die Gefahrenstandorte sind in der Tabelle HAZARDOUS_SITES gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wird. Die als Punkt definierte Spalte LOCATION speichert einen Standort, der das geographische Zentrum jedes Gefahrenstandorts darstellt.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

Die Funktion buffer generiert einen Fünf-Meilen-Puffer um die Standorte der Lagerstätten für gefährlichen Abfall. Die Funktion ST_Intersection generierte Polygone aus der Schnittmenge der gepufferten Polygone um die Abfallstandorte und der sensiblen Bereiche. Die Funktion ST_Area gibt die Polygonfläche der Schnittmenge zurück; diese wird mit der Funktion SUM als Summe über alle Gefahrenstandorte gebildet. Die Klausel GROUP BY weist die Abfrage an, die Schnittmengenflächen nach site_ID der Gefahrenstandorte zu akkumulieren.

```
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;
```

In Abb. 35 stehen die Kreise für die gepufferten Polygone um die Gefahrenstandorte. Die Schnittmenge dieser Pufferpolygone mit den Polygonen der sensiblen Bereiche erzeugen drei weitere Polygone. Das Krankenhaus links oben wird zwei Mal geschnitten; die Schule rechts unten nur ein Mal.

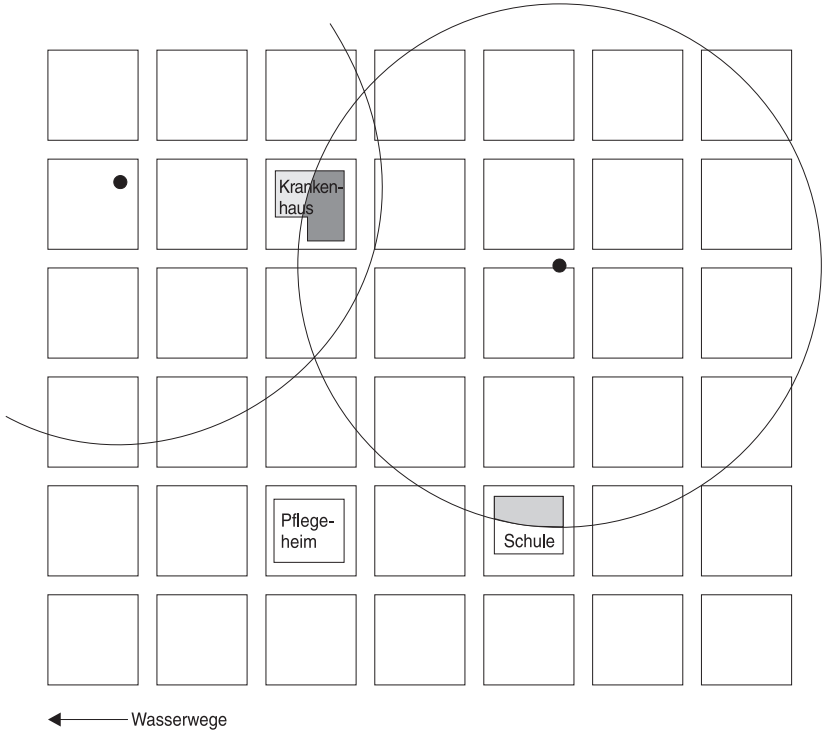


Abbildung 35. Mit ST_Intersection feststellen, wie groß der gefährdete Bereich in den Gebäuden ist

ST_Intersects

ST_Intersects verwendet zwei Geometrieobjekte und gibt 1 (TRUE) zurück, wenn die Schnittmenge der beiden Geometrien keine leere Menge ergibt. Andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_Intersects(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Der Chef der Feuerwehr benötigt eine Liste aller sensiblen Bereiche innerhalb eines Radius von 5 Meilen um die Lagerstätten für gefährlichen Abfall.

Die sensiblen Bereiche sind in der Tabelle SENSITIVE_AREAS gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wird. Die Spalte ZONE ist als Polygon definiert, in dem der Umriß der jeweiligen sensiblen Bereiche gespeichert ist.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

Die Gefahrenstandorte sind in der Tabelle HAZARDOUS_SITES gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wird. Die als Punkt definierte Spalte LOCATION speichert einen Standort, der das geographische Zentrum jedes Gefahrenstandorts darstellt.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

Die Abfrage gibt eine Liste der sensiblen Bereiche und der Namen der Gefahrenstandorte zurück, sie sich mit dem Fünf-Meilen-Puffer um die Gefahrenstandorte schneiden.

```
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 verwendet eine Linienfolge oder eine Mehrlinienfolge und gibt 1 (TRUE) zurück, falls sie geschlossen ist; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_IsClosed(c db2gse.ST_Curve)
db2gse.ST_IsClosed(mc db2gse.ST_MultiCurve)

Rückgabetyp

Integer

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle CLOSED_LINESTRING mit einer einzigen Linienspalte.

```
CREATE TABLE CLOSED_LINESTRING (ln1 db2gse.ST_LineString)
```

Die folgenden INSERT-Anweisungen fügen zwei Datensätze in die Tabelle CLOSED_LINESTRING ein. Der erste Datensatz ist keine geschlossene Linie, der zweite Datensatz dagegen schon.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe zeigen die Ergebnisse der Funktion ST_IsClosed. Die erste Zeile gibt 0 zurück, weil die Linienfolge nicht geschlossen ist; die zweite Zeile gibt 1 zurück, weil die Linienfolge geschlossen ist.

```
SELECT db2gse.ST_IsClosed(ln1) "Is it closed" FROM CLOSED_LINESTRING

Is it closed
------------
0
1

  2 record(s) selected.
```

Die folgende Anweisung CREATE TABLE erstellt die Tabelle CLOSED_MULTILINESTRING mit einer einzigen Mehrlinienspalte.

```
CREATE TABLE CLOSED_MULTILINESTRING (mln1 db2gse.ST_MultiLineString)
```

Die folgenden INSERT-Anweisungen fügen zwei Datensätze in CLOSED-_MULTILINESTRING ein, einen Datensatz mit einer nicht geschlossenen Mehrlinienfolge und einen Datensatz mit einer geschlossenen Mehrlinienfolge.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe zeigen die Ergebnisse der Funktion ST_IsClosed. Die erste Zeile gibt 0 zurück, weil die Mehrlinienfolge nicht geschlossen ist; die zweite Zeile gibt 1 zurück, weil die Mehrlinienfolge geschlossen ist. Eine Mehrlinienfolge gilt als geschlossen, wenn alle ihre Linienfolgen geschlossen sind.

```
SELECT db2gse.ST_IsClosed(mln1) "Is it closed" FROM CLOSED_MULTILINESTRING

Is it closed
------------
0
1

  2 record(s) selected.
```

ST_IsEmpty

ST_IsEmpty verwendet ein Geometrieobjekt und gibt 1 (TRUE) zurück, falls es leer ist; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_IsEmpty(g db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle EMPTY_TEST mit zwei Spalten. Die Spalte GEOTYPE speichert den Datentyp der in der Geometriespalte G1 gespeicherten Unterklassen.

```
CREATE TABLE EMPTY_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Die folgenden INSERT-Anweisungen fügen zwei Datensätze für die Geometrieunterklassen Punkt, Linienfolge und Polygon ein. Ein Datensatz ist leer, der andere nicht.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe zeigen den Geometrietyp der Spalte GEOTYPE und die Ergebnisse der Funktion 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 verwendet eine Linienfolge und gibt 1 (TRUE) zurück, wenn es sich um einen Ring handelt (die Linienfolge also geschlossen und einfach ist); andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_IsRing(c db2gse.ST_Curve)

Rückgabetyp

Integer

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle RING_LINESTRING mit einer einzigen Linienspalte mit dem Namen LN1.

```
CREATE TABLE RING_LINESTRING (ln1 db2gse.ST_LineString)
```

Die folgenden INSERT-Anweisungen fügen drei Linienfolgen in die Spalte LN1 ein. Die erste Zeile enthält eine Linienfolge, die nicht geschlossen ist und somit keinen Ring darstellt. Die zweite Zeile enthält eine Linienfolge, die geschlossen ist und somit einen Ring darstellt. Die dritte Zeile enthält eine geschlossene Linienfolge, die jedoch nicht einfach ist, da sie ihren eigenen Innenbereich schneidet; sie stellt somit keinen Ring dar.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe zeigen die Ergebnisse der Funktion ST_IsRing. Die erste und dritte Zeile geben jeweils 0 zurück, weil ihre Linienfolgen keine Ringe sind; die zweite Zeile gibt dagegen 1 zurück, weil es sich dabei um einen Ring handelt.

```
SELECT db2gse.ST_IsRing(ln1) "Ring" FROM RING_LINESTRING

Ring
-----------
0
1
0

  3 record(s) selected.
```

ST_IsSimple

ST_IsSimple verwendet ein Geometrieobjekt und gibt 1 (TRUE) zurück, wenn das Objekt einfach (″simple″) ist; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_IsSimple(g db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle ISSIMPLE_TEST mit zwei Spalten. Die Spalte PID, ein smallint-Element, enthält die eindeutige Kennung für jede Zeile. Die Geometriespalte G1 speichert die einfachen und nicht einfachen Geometriemuster.

```
CREATE TABLE ISSIMPLE_TEST (pid smallint, g1 db2gse.ST_Geometry)
```

Die folgenden INSERT-Anweisungen fügen zwei Datensätze in die Tabelle ISSIMPLE_TEST ein. Der erste Datensatz ist einfach, weil es sich um eine Linienfolge handelt, die ihren Innenbereich nicht schneidet. Der zweite Datensatz ist nicht einfach, weil er seinen Innenbereich schneidet.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe zeigen die Ergebnisse der Funktion ST_IsSimple. Der erste Datensatz gibt 1 zurück, weil die Linienfolge einfach ist; der zweite Datensatz gibt 0 zurück, weil die Linienfolge nicht einfach ist.

```
SELECT ST_IsSimple(g1)
FROM ISSIMPLE_TEST

g1
--------------
1
0
```

ST_IsValid

ST_IsValid verwendet eine ST_Geometry und gibt 1 (TRUE) zurück, falls es gültig ist; andernfalls wird 0 (FALSE) zurückgegeben Eine in eine DB2-Datenbank eingefügte Geometrie ist immer gültig, weil der DB2 Spatial Extender ihre räumlichen Daten vor dem Akzeptieren immer überprüft. Andere DBMS-Anbieter überprüfen die Eingabe jedoch unter Umständen nicht, sondern erwarten, daß die Anwendung dies tut.

Syntax

db2gse.ST_IsValid(g db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die Tabelle valid_test wird mit den Spalten geotype und g1 erstellt. Die Spalte geotype speichert den Namen der in der Geometriespalte g1 gespeicherten Geometrieunterklasse.

```
CREATE TABLE valid_test (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Die INSERT-Anweisungen fügen eine Muster-Unterklasse in die Tabelle valid_test ein.

```
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))
)
```

Die Anweisung SELECT listet den Unterklassennamen auf, der in der Spalte geotype mit der Dimension dieses Geotyps gespeichert ist.

```
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 verwendet eine Linienfolge oder eine Mehrlinienfolge und gibt ihre Länge zurück.

Syntax

db2gse.ST_Length(c db2gse.ST_Curve)
db2gse.ST_Length(mc db2gse.ST_MultiCurve)

Rückgabetyp

Double

Beispiele

Ein Heimatökologe untersucht das Wanderverhalten des Lachsbestands in den Wasserläufen des Landkreises. Der Ökologe möchte die Länge aller Bäche und Flüsse im Landkreis wissen.

Die folgende Anweisung CREATE TABLE erstellt die Tabelle WATERWAYS. Die Spalten ID und NAME kennzeichnen alle in der Tabelle gespeicherten Bäche und Flüsse. Die Spalte WATER ist eine Mehrlinienfolge, da der Fluß und die Flußsysteme häufig eine Zusammenfassung verschiedener Linienfolgen sind.

```
CREATE TABLE WATERWAYS (id integer, name varchar(128),
water db2gse.ST_MultiLineString);
```

Die folgenden Anweisung SELECT verwendet die Funktion ST_Length, um den Namen und die Länge jedes Wasserweges im Landkreis zurückzugeben.

```
SELECT name, db2gse.ST_Length(water) "Length"
FROM WATERWAYS;
```

Abb. 36 zeigt das Fluß- und Bachsystem innerhalb der Grenzen des Landkreises.

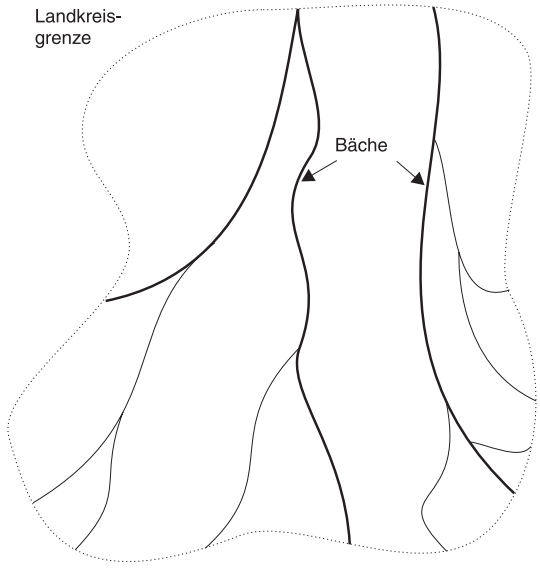


Abbildung 36. Mit ST_Length die Gesamtlänge der Wasserwege im Landkreis ermitteln

ST_LineFromText

ST_LineFromText verwendet eine bekannte Textdarstellung des Typs Linienfolge und die Identität eines räumlichen Bezugssystems und gibt eine Linienfolge zurück.

Syntax

db2gse.ST_LineFromText(lineStringTaggedText Varchar(4000), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_LineString

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle LINESTRING_TEST mit einer einzigen Linienspalte LN1.

```
CREATE TABLE LINESTRING_TEST (ln1 db2gse.ST_LineString)
```

Die folgende Anweisung INSERT fügt über die Funktion ST_LineFromText eine Linienfolge in die Spalte LN1 ein.

```
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 verwendet eine bekannte binäre Darstellung des Typs Linienfolge und die Identität eines räumlichen Bezugssystems und gibt eine Linienfolge zurück.

Syntax

db2gse.ST_LineFromWKB(WKBLineString Blob(1M), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_LineString

Beispiele

Der folgende Codeabschnitt füllt die Tabelle SEWERLINES mit der eindeutigen ID, der Größenklasse und der Geometrie aller Abwasserkanallinien auf.

Die Tabelle SEWERLINES wird mit drei Spalten erstellt. Die erste Spalte, SEWER_ID, kennzeichnet jede Abwasserkanallinie eindeutig. Die zweite Spalte CLASS des Typs Integer kennzeichnet den Typ des Abwasserkanals; normalerweise gibt diese Spalte die Kapazität des Kanals an. Die dritte Spalte, SEWER, des Typs Linienfolge speichert die Geometrie der Abwasserkanallinie.

```
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 verwendet eine bekannte Textdarstellung des Typs Mehrlinienfolge und die Identität eines räumlichen Bezugssystems und gibt eine Mehrlinienfolge zurück.

Syntax

db2gse.ST_MLineFromText(multiLineStringTaggedText String, cr db2gse.coordref)

Rückgabetyp

db2gse.ST_MultiLineString

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle MLINESTRING_TEST. MLINESTRING_TEST hat zwei Spalten: die smallint-Spalte GID, die die Zeile eindeutig kennzeichnet, und die Mehrlinienspalte ML1.

```
CREATE TABLE ST_MLINESTRING_TEST (gid smallint, ml1 db2gse.ST_MultiLineString)
```

Die folgende Anweisung INSERT fügt die Mehrlinienfolge mit der Funktion ST_MLineFromText ein.

```
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 verwendet eine bekannte binäre Darstellung des Typs Mehrlinienfolge und die Identität eines räumlichen Bezugssystems und gibt eine Mehrlinienfolge zurück.

Syntax

db2gse.ST_MLineFromWKB(WKBMultiLineString Blob(1M), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_MultiLineString

Beispiele

Der folgende Codeabschnitt füllt die Tabelle WATERWAYS mit einer eindeutigen ID, einem Namen und einer Mehrlinienfolge WATER aus.

Die Tabelle WATERWAYS wird erstellt mit den Spalten ID und NAME, die jeden in der Tabelle gespeicherten Fluß und Wasserlauf kennzeichnen. Die Spalte WATER ist eine Mehrlinienfolge, da der Fluß und die Flußsysteme häufig eine Zusammenfassung verschiedener Linienfolgen sind.

```
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 verwendet eine bekannte Textdarstellung des Typs Multipoint (Mehrpunkt) und die Identität eines räumlichen Bezugssystems und gibt eine Mehrpunktangabe zurück.

Syntax

db2gse.ST_MPointFromText(multiPointTaggedText Varchar(4000), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_MultiPoint

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle MULTIPOINT_TEST mit einer einzigen Mehrpunktspalte MPT1.

```
CREATE TABLE MULTIPOINT_TEST (mpt1 db2gse.ST_MultiPoint)
```

Die folgende Anweisung INSERT fügt über die Spalte ST_MPointFromText eine Mehrpunktangabe in die Spalte MPT1 ein.

```
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 verwendet eine bekannte binäre Darstellung des Typs Multipoint (Mehrpunkt) und die Identität eines räumlichen Bezugssystems und gibt eine Mehrpunktangabe zurück.

Syntax

db2gse.ST_MPointFromWKB(WKBMultiPoint Blob(1M), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_MultiPoint

Beispiele

Der folgende Codeabschnitt füllt die Tabelle SPECIES_SITINGS aus.

Die Tabelle SPECIES_SITINGS wird mit drei Spalten erstellt. Die Spalten SPECIES und GENUS kennzeichnen jede Zeile eindeutig, während die Mehrpunktangabe SITINGS die Standorte der Spezies speichert.

```
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 verwendet eine bekannte Textdarstellung des Typs Multipolygon und die Identität eines räumlichen Bezugssystems und gibt ein Multipolygon zurück.

Syntax

db2gse.ST_MPolyFromText(multiPolygonTaggedText Varchar(4000), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_MultiPolygon

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle MULTIPOLYGON_TEST mit einer einzigen Multipolygonspalte MPL1.

```
CREATE TABLE MULTIPOLYGON_TEST (mpl1 db2gse.ST_MultiPolygon)
```

Die folgende Anweisung INSERT fügt über die Funktion ST_MPolyFromText ein Multipolygon in die Spalte MPL1 ein.

```
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 verwendet eine bekannte binäre Darstellung des Typs Multipolygon und die Identität eines räumlichen Bezugssystems und gibt ein Multipolygon zurück.

Syntax

db2gse.ST_MPolyFromWKB(WKBMultiPolygon Blob(1M), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_MultiPolygon

Beispiele

Der folgende Codeabschnitt füllt die Tabelle LOTS aus.

Die Tabelle LOTS speichert die LOT_ID, die jedes Grundstück eindeutig kennzeichnet und das Multipolygon für das Grundstück, die die Geometrie für die Grundstückslinie enthält.

```
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 verwendet eine Gruppe und gibt die Anzahl der Geometrien in der Gruppe zurück.

Syntax

db2gse.ST_NumGeometries(g db2gse.ST_GeomCollection)

Rückgabetyp

Integer

Beispiele

Der Stadtplaner muß die tatsächliche Anzahl der Einzelgebäude zu jeder Gebäudegrundfläche kennen.

Die Gebäudegrundflächen sind in der Tabelle BUILDINGFOOTPRINTS gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wurde.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

Die folgende Anweisung SELECT verwendet die Funktion ST_NumGeometries zum Auflisten der BUILDING_ID, die jedes Gebäude und die Anzahl der Gebäude für jede Grundfläche eindeutig kennzeichnet.

```
SELECT building_id, db2gse.ST_NumGeometries (footprint) "Number of buildings"
FROM BUILDINGFOOTPRINTS;
```

ST_NumInteriorRing

ST_NumInteriorRing verwendet ein Polygon und gibt die Anzahl seiner inneren Ringe zurück.

Syntax

db2gse.NumInteriorRing(p db2gse.ST_Polygon)

Rückgabetyp

Integer

Beispiele

Eine Ornithologin, die den Vogelbestand auf verschiedenen Südseeinseln untersuchen möchte, weiß, daß sich die Zone für die Nahrungssuche einer bestimmten Spezies auf die Inseln mit Süßwasserseen beschränkt. Sie möchte daher wissen, welche Inseln ein oder mehrere Seen haben.

Die folgende Anweisung CREATE TABLE erstellt die Tabelle ISLANDS. Die Spalten ID und NAME der Tabelle ISLANDS kennzeichnen alle Inseln, und die Polygonspalte LAND speichert die Geometrie dieser Inseln.

```
CREATE TABLE ISLANDS (id integer, name varchar(32), land db2gse.ST_Polygon);
```

Da die inneren Ringe die Seen darstellen, werden mit der Funktion ST_NumInteriorRing nur die Inseln aufgelistet, die mindestens einen inneren Ring haben.

```
SELECT name FROM ISLANDS WHERE db2gse.ST_NumInteriorRing(land) > 0;
```

ST_NumPoints

ST_NumPoints verwendet eine Linienfolge und gibt die Anzahl ihrer Punkte zurück.

Syntax

db2gse.ST_NumPoints(l db2gse.ST_LineString)

Rückgabetyp

Integer

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle NUMPOINTS_TEST. Die Spalte GEOTYPE enthält den in der Geometriespalte G1 gespeicherten Geometrietyp.

```
CREATE TABLE NUMPOINTS_TEST (geotype varchar(12), g1 db2gse.ST_Geometry)
```

Die folgende Anweisung INSERT fügt eine Linienfolge ein.

```
INSERT INTO NUMPOINTS_TEST VALUES( linestring,
db2gse.ST_LineFromText('linestring (10.02 20.01, 23.73 21.92)',
db2gse.coordref()..srid(0)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe listet den Geometrietyp und die Anzahl der darin enthaltenen Punkte auf.

```
SELECT geotype, db2gse.ST_NumPoints(g1)
FROM NUMPOINTS_TEST

GEOTYPE      Number of points
------------ ----------------
ST_linestring          2
1 record(s) selected.
```

ST_OrderingEquals

ST_OrderingEquals vergleicht zwei Geometrien und gibt 1 (TRUE) zurück, wenn die Geometrien gleich sind und die Koordinaten die gleiche Reihenfolge haben; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_OrderingEquals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle LINESTRING-_TEST mit den beiden Linienspalten L1 und L2.

```
CREATE TABLE LINESTRING_TEST (lid integer, l1 db2gse.ST_LineString,
l2 db2gse.ST_LineString);
```

Die folgende Anweisung INSERT fügt zwei Linienfolgen ein in L1 und L2, die gleich sind und dieselbe Koordinatenreihenfolge haben.

```
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)));
```

Die folgende Anweisung INSERT fügt zwei Linienfolgen ein in L1 und L2, die gleich sind, aber nicht dieselbe Koordinatenreihenfolge haben.

```
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)));
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe zeigt, wie die Funktion ST_Equals unabhängig von der Reihenfolge der Koordinaten 1 (TRUE) zurückgibt. Die Funktion ST_OrderingEquals gibt 0 (FALSE) zurück, wenn die Geometrien nicht gleich sind und die gleiche Koordinatenreihenfolge haben.

```
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 verwendet zwei Geometrieobjekte und gibt 1 (TRUE) zurück, wenn die Schnittmenge der beiden Objekte ein Geometrieobjekt derselben Dimension ergibt, das jedoch mit keinem der Quellenobjekte identisch ist; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_Overlaps(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Der zuständige Planer benötigt eine Liste der Abfall-Gefahrenstandorte, in deren Umkreis von fünf Meilen sensible Bereiche wie Schulen, Krankenhäuser und Pflegeheime liegen.

Die folgende Anweisung CREATE TABLE erstellt die Tabelle SENSITIVE_AREAS. Die Tabelle SENSITIVE_AREAS enthält verschiedene Spalten, die die gefährdeten Institutionen beschreiben, zusätzlich zu der Spalte ZONE, die die Polygongeometrie der Institution beschreibt.

```
CREATE TABLE SENSITIVE_AREAS (id       integer,
                              name     varchar(128),
                              size     float,
                              type     varchar(10),
                              zone     db2gse.ST_Polygon);
```

In der Tabelle HAZARDOUS_SITES ist die Identität der Standorte in den Spalten SITE_ID und NAME gespeichert; der tatsächliche geographische Standort wird jeweils in der Punktspalte LOCATION gespeichert.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

In der folgenden Anweisung SELECT werden die Tabellen SENSITIVE_AREAS and HAZARDOUS_SITES über die Funktion ST_Overlaps verknüpft. Die Funktion gibt 1 (TRUE) zurück für alle Zeilen in der Tabelle SENSITIVE_AREAS, deren Zonenpolygone den gepufferten Fünf-Meilen-Radius um den HAZARDOUS_SITES-Standort überlappen.

```
SELECT hs.name
FROM HAZARDOUS_SITES hs, SENSITIVE_AREAS sa
WHERE db2gse.ST_Overlaps (buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

In Abb. 37 überlappen das Krankenhaus und die Schule den Fünf-Meilen-Radius der beiden Abfallstandorte des Landkreises; das Pflegeheim liegt außerhalb dieses Radius.

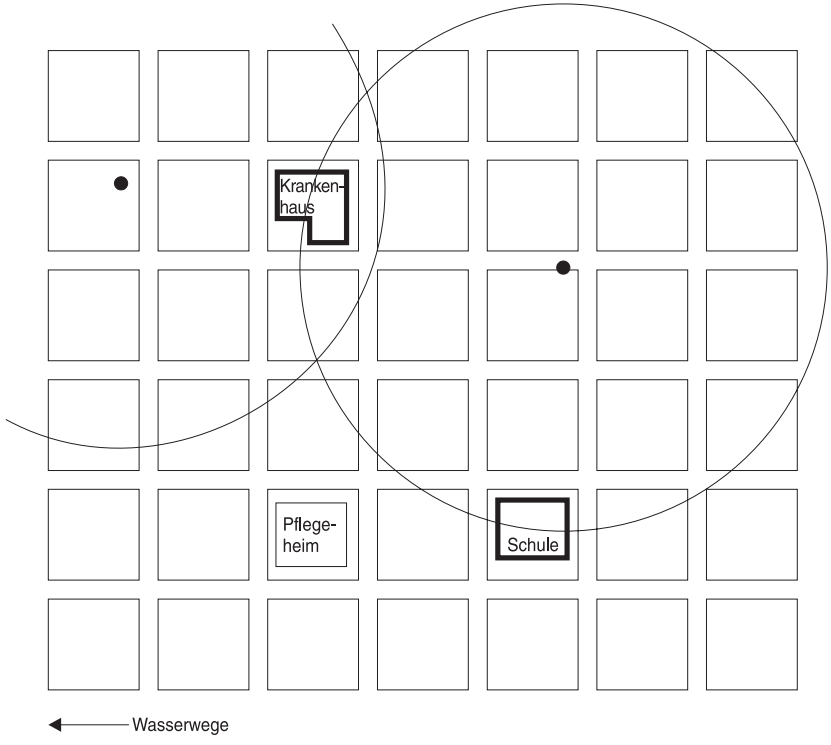


Abbildung 37. Mit ST_Overlaps die Gebäude ermitteln, die zumindest teilweise im Gefährdungsbereich der Abfall-Lagerstätten liegen

ST_Perimeter

ST_Perimeter gibt den Umfang eines ST_Surface zurück.

Syntax

db2gse.ST_Perimeter(s db2gse.ST_Surface)
db2gse.ST_Perimeter(ms db2gse.ST_MultiSurface)

Rückgabetyp

Double

Beispiele

Ein Ornithologe, der das Verhalten der Vögel an den Ufern von Seen studiert, muß die Uferlinie der Seen innerhalb eines bestimmten Gebiets kennen. Die Seen sind als Multipolygone in der Tabelle WATERBODIES gespeichert; diese Tabelle wurde mit der folgenden Anweisung CREATE TABLE erstellt.

```
CREATE TABLE WATERBODIES (wbid  integer,
                          waterbody  db2gse.ST_MultiPolygon);
```

In der folgenden Anweisung SELECT gibt die Funktion ST_Perimeter den Umfang jeder Wasserfläche zurück; die Funktion SUM akkumuliert die Umfangswerte und gibt ihre Summe zurück.

```
SELECT SUM(db2gse.ST_Perimeter(waterbody))
FROM waterbodies;
```

ST_PointFromText

ST_PointFromText verwendet eine bekannte Textdarstellung des Typs Punkt und die Identität eines räumlichen Bezugssystems und gibt einen Punkt zurück.

Syntax

db2gse.ST_PointFromText(pointTaggedText Varchar(4000), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_Point

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle POINT_TEST mit einer einzigen Punktspalte PT1.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

Bevor die Anweisung INSERT den Punkt in die Spalte PT1 einfügt, wandelt die Funktion ST_PointFromText die Punkttextkoordinaten in das Punktformat um.

```
INSERT INTO POINT_TEST VALUES (
       db2gse.ST_PointFromText ('point(10.01 20.03)',
         db2gse.coordref()..srid(0)))
```

ST_PointFromWKB

ST_PointFromWKB verwendet eine bekannte binäre Darstellung des Typs Punkt und die Identität eines räumlichen Bezugssystems und gibt einen Punkt zurück.

Syntax

db2gse.ST_PointFromWKB(WKBPoint Blob(1M), srs SRID)

Rückgabetyp

db2gse.ST_Point

Beispiele

Der folgende Codeabschnitt füllt die Tabelle HAZARDOUS_SITES aus.

Die Gefahrenstandorte sind in der Tabelle HAZARDOUS_SITES gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wird. Die als Punkt definierte Spalte LOCATION speichert einen Standort, der das geographische Zentrum jedes Gefahrenstandorts darstellt.

```
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 gibt einen ST_Point zurück; als Eingabe verwendet die Funktion eine X-Koordinate, eine Y-Koordinate und einen räumlichen Bezug.

Syntax

db2gse.ST_Point(X Double, Y Double, srs SRID)

Rückgabetyp

db2gse.ST_Point

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle POINT_TEST mit einer einzigen Punktspalte PT1.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

Die Funktion ST_Point wandelt die Punktkoordinaten in eine Punktgeometrie um, bevor die Anweisung INSERT sie in die Spalte PT1 einfügt.

```
INSERT INTO point_test VALUES(
    db2gse.ST_Point(10.01,20.03, db2gse.coordref()..srid(0))
)
```

ST_PointN

ST_PointN verwendet eine Linienfolge und einen ganzzahligen Index und gibt einen Punkt zurück, der den n-ten Scheitelpunkt im Verlauf der Linienfolge darstellt.

Syntax

db2gse.ST_PointN(l db2gse.ST_Curve, n Integer)

Rückgabetyp

db2gse.ST_Point

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle POINTN_TEST mit zwei Spalten: der Spalte GID, die jede Zeile eindeutig kennzeichnet, und der Linienspalte LN1.

```
CREATE TABLE POINTN_TEST (gid integer, ln1 db2gse.ST_LineString)
```

Die folgenden INSERT-Anweisungen fügen zwei Linienwerte ein. Die erste Zeile hat keine Z-Koordinaten oder Maßangaben; die zweite dagegen schon.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe listet die Spalte GID und den zweiten Scheitelpunkt jeder Linienfolge auf. Die erste Zeile gibt einen Punkt zurück, der weder Z-Koordinate noch ein Maß enthält; der aus der zweiten Zeile resultierende Punkt enthält eine Z-Koordinate und ein Maß. Die Funktion ST_PointN gibt Punkte mit einer Z-Koordinate oder einem Maß zurück, falls diese in der Quellenlinie enthalten sind.

```
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 verwendet ein Polygon oder ein Multipolygon und gibt einen ST_Point zurück.

Syntax

db2gse.ST_PointOnSurface(s db2gse.ST_Surface)
db2gse.ST_PointOnSurface(ms db2gse.ST_MultiSurface)

Rückgabetyp

db2gse.ST_Point

Beispiele

Der Stadtplaner muß einen Kennzeichnungspunkt für jede Gebäudegrundfläche erstellen.

Die Gebäudegrundflächen sind in der Tabelle BUILDINGFOOTPRINTS gespeichert, die mit der folgenden Anweisung CREATE TABLE erstellt wurde.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

Die Funktion ST_PointOnSurface generiert einen Punkt, der garantiert auf der Gebäudegrundfläche liegt. Die Funktion ST_PointOnSurface gibt einen Punkt zurück, der von der Funktion AsBinaryShape in eine 1-Megabyte-Zeichenfolge umgewandelt wird; diese Zeichenfolge kann von der Anwendung verwendet werden.

```
SELECT CAST(db2gse.AsBinaryShape(db2gse.ST_PointOnSurface(footprint)) as
            blob(1m))
FROM BUILDINGFOOTPRINTS;
```

ST_PolyFromText

ST_PolyFromText verwendet eine bekannte Textdarstellung des Typs Polygon und die Identität eines räumlichen Bezugssystems und gibt ein Polygon zurück.

Syntax

db2gse.ST_PolyFromText(polygonTaggedText Varchar(4000), cr db2gse.coordref)

Rückgabetyp

db2gse.ST_Polygon

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle POLYGON_TEST mit einer einzigen Polygonspalte.

```
CREATE TABLE POLYGON_TEST (pl1 db2gse.ST_Polygon)
```

Die folgende Anweisung INSERT fügt über die Funktion ST_PolyFromText ein Polygon in die Polygonspalte ein.

```
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 verwendet eine bekannte binäre Darstellung des Typs Polygon und die Identität eines räumlichen Bezugssystems und gibt ein Polygon zurück.

Syntax

db2gse.ST_PolyFromWKB(WKBPolygon Blob(1M), SRID Integer)

Rückgabetyp

db2gse.ST_Polygon

Beispiele

Der folgende Codeabschnitt füllt die Tabelle SENSITIVE_AREAS aus.

Die Tabelle SENSITIVE_AREAS enthält verschiedene Spalten, die die gefährdeten Institutionen beschreiben, zusätzlich zu der Spalte zone, die die Polygongeometrie der Institution beschreibt.

```
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 generiert ein ST_Polygon und eine Kennung eines räumlichen Bezugssystems aus einem ST_LineString.

Syntax

db2gse.ST_Polygon(l db2gse.ST_LineString, cr db2gse.coordref)

Rückgabetyp

db2gse.ST_Polygon

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle POLYGON_TEST mit einer einzigen Spalte, P1.

```
CREATE TABLE POLYGON_TEST (p1 db2gse.ST_polygon)
```

Die folgende Anweisung INSERT wandelt einen Ring (eine geschlossene, einfache Linienfolge) in ein Polygon um und fügt Sie mit der Funktion ST_LineFromText innerhalb der Funktion ST_Polygon in die Spalte P1 ein.

```
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 vergleicht zwei Geometrien und gibt 1 (TRUE) zurück, wenn die Geometrien die von der Mustermatrix-Zeichenfolge DE-9IM definierten Bedingungen erfüllen; andernfalls wird 0 (FALSE) zurückgegeben. Informationen zu der DE-9IM-Mustermatrix finden Sie im Abschnitt „Prädikatfunktionen" auf Seite 145.

Syntax

db2gse.ST_Relate(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry, patternMatrix String)

Rückgabetyp

Integer

Beispiele

Eine DE-9IM-Mustermatrix ist eine Einheit zum Vergleich von Geometrien. Es gibt verschiedene Arten solcher Matrizen. Die Mustermatrix *equals* gibt beispielsweise an, ob zwei Geometrien gleich sind.

In diesem Beispiel wird eine "equals"-Mustermatrix (siehe Tabelle 56) von links nach rechts und von oben nach unten in eine Zeichenfolge („T*F**FFF*") eingelesen.

Tabelle 56. Equals-Mustermatrix

		b		
		Innenbereich	**Begrenzung**	**Außenbereich**
	Innenbereich	T	*	F
a	**Begrenzung**	*	*	F
	Außenbereich	F	F	*

Als nächstes wird die Tabelle RELATE_TEST mit der folgenden Anweisung CREATE TABLE erstellt.

```
CREATE TABLE RELATE_TEST (rid integer, g1 db2gse.ST_Geometry,
g2 db2gse.ST_Geometry, g3 db2gse.ST_Geometry);
```

Die folgenden INSERT-Anweisungen fügen eine Muster-Unterklasse in die Tabelle RELATE_TEST ein.

```
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)
)
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe listet den Unterklassennamen auf, der in der Spalte GEOTYPE mit der Dimension dieses Geotyps gespeichert ist.

```
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 verwendet ein Geometrieobjekt und gibt die Identität seines räumlichen Bezugssystems zurück.

Syntax

db2gse.ST_SRID(g1 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Bei der Installation des DB2 Spatial Extender wird die Tabelle SPATIAL_REFERENCES erstellt. Beim Erstellen einer Geometrie wird die SRID dieser Geometrie in die Tabelle SPATIAL_REFERENCES eingetragen. Die Funktion ST_SRID gibt den Wert dieses Eintrags zurück.

In einer Anweisung CREATE TABLE wird beispielsweise ein Geometrietyp verwendet:

```
CREATE TABLE SRID_TEST(g1 db2gse.ST_Geometry)
```

In der folgenden Anweisung INSERT wird eine Punktegeometrie an der Koordinatenposition 10.01,50.76 in die Geometriespalte G1 eingetragen. Wenn die Punktgeometrie von der Funktion ST_PointFromText erstellt wurde, so hat sie den srid-Wert 1 erhalten.

```
INSERT INTO SRID_TEST
VALUES (db2gse.ST_PointFromText('point(10.01 50.76)',
    db2gse.coordref()..srid(0)))
```

Die Funktion ST_SRID gibt die Identität des räumlichen Bezugssystems der eben eingegebenen Geometrie zurück. Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe verdeutlicht dies.

```
SELECT db2gse.ST_SRID(g1) FROM SRID_TEST

g1
--------------
1
```

ST_StartPoint

ST_StartPoint verwendet eine Linienfolge und gibt einen Punkt zurück, der den ersten Punkt der Linienfolge darstellt.

Syntax

db2gse.ST_StartPoint(c db2gse.ST_Curve)

Rückgabetyp

db2gse.ST_Point

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle STARTPOINT_TEST. STARTPOINT_TEST enthält zwei Spalten: die Spalte GID, die die Zeilen der Tabelle eindeutig kennzeichnet, und die Linienspalte LN1.

```
CREATE TABLE STARTPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

Die folgenden INSERT-Anweisungen fügen die Linienfolgen in die Spalte LN1 ein. Die erste Zeile hat keine Z-Koordinaten oder Maßangaben; die zweite dagegen schon.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe zeigen, wie die Funktion ST_StartPoint den ersten Punkt jeder Linienfolge extrahiert. Die Funktion ST_AsText wandelt den Punkt in sein Textformat um. Der erste Punkt in der Liste hat keine Z-Koordinate und kein Maß; der zweite Punkt hat beides, weil diese Elemente auch in der Quellenlinie enthalten sind.

```
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 verwendet zwei Geometrieobjekte und gibt ein Geometrieobjekt zurück, das die symmetrische Differenz der Eingabeobjekte darstellt.

Syntax

db2gse.ST_SymmetricDiff(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Der zuständige Planer muß den Bereich der sensiblen Bereiche und des Fünf-Meilen-Radius ohne Schnittmenge ermitteln.

Die folgende Anweisung CREATE TABLE erstellt die Tabelle SENSITIVE_AREAS, die verschiedene Spalten zur Beschreibung der gefährdeten Institutionen enthält. Die Tabelle SENSITIVE_AREAS enthält auch die Spalte ZONE, in der die Polygongeometrie der Institution gespeichert ist.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

Die folgende Anweisung CREATE TABLE erstellt die Tabelle HAZARDOUS_SITES, in der die Identität der Standorte in den Spalten SITE_ID und NAME gespeichert wird; der tatsächliche geographische Standort jeder Site wird in der Punktspalte LOCATION gespeichert.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  point);
```

Die Funktion ST_Buffer generiert einen Fünf-Meilen-Puffer um die Standorte der Lagerstätten für gefährlichen Abfall. Die Funktion ST_SymmetricDiff generierte Polygone aus der Schnittmenge der gepufferten Polygone um die Abfallstandorte und der sensiblen Bereiche. Die Funktion ST_Area gibt die Polygonfläche der Schnittmenge für jeden Gefahrenstandort zurück.

```
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
```

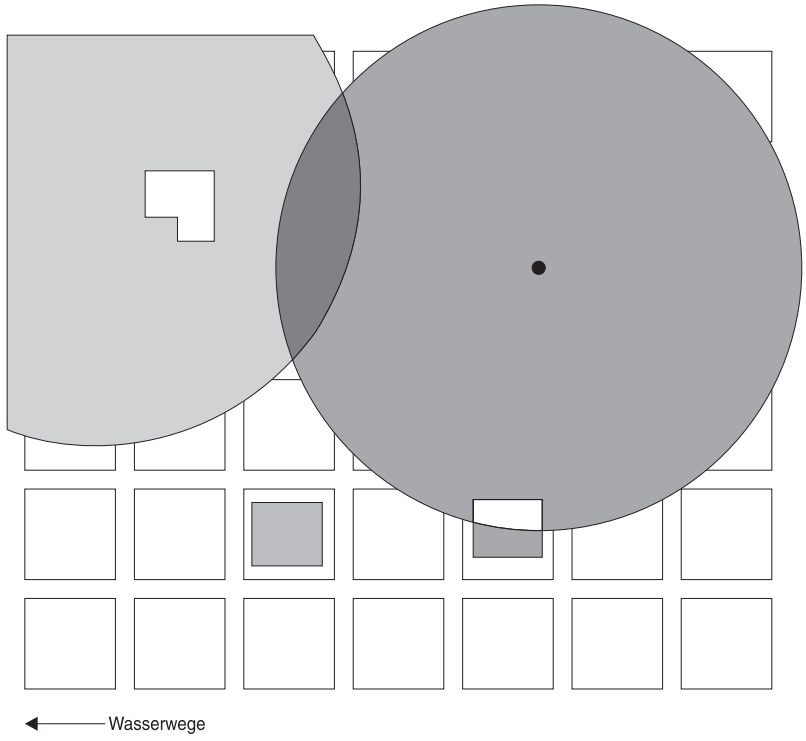


Abbildung 38. Mit ST_SymmetricDiff die Gefahrenbereiche ermitteln, die keine sensiblen Bereiche (bewohnte Gebäude) enthalten

In Abb. 38 führt die symmetrische Differenz aus Gefahrengebieten und den sensiblen Bereichen zur Subtraktion der Schnittbereiche.

ST_Touches

ST_Touches gibt 1 (TRUE) zurück, wenn keiner der gemeinsamen Punkte der beiden Geometrien den Innenbereich beider Geometrien schneidet; andernfalls wird 0 (FALSE) zurückgegeben. Mindestens eine der Geometrien muß eine Linienfolge, ein Polygon, eine Mehrlinienfolge oder ein Multipolygon sein.

Syntax

db2gse.ST_Touches(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Der GIS-Techniker muß eine Liste aller Abwasserkanäle, deren Endpunkt einen anderen Abwasserkanal schneiden, zur Verfügung stellen.

Die folgende Anweisung CREATE TABLE erstellt die Tabelle SEWERLINES mit drei Spalten. Die erste Spalte, SEWER_ID, kennzeichnet jede Abwasserkanallinie eindeutig. Die zweite Spalte CLASS des Typs Integer kennzeichnet den Typ des Abwasserkanals; normalerweise gibt diese Spalte die Kapazität des Kanals an. Die dritte Spalte, SEWER, des Typs Linienfolge speichert die Geometrie der Abwasserkanallinie.

```
CREATE TABLE SEWERLINES (sewer_id integer, class integer, sewer
    db2gse.ST_LineString);
```

Die folgende Anweisung SELECT gibt eine sortierte Liste der SEWER_IDS, die einander berühren, zurück.

```
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 ordnet eine Geometrie einem anderen räumlichen Bezugssystem zu als dem momentan zugeordneten.

Syntax

db2gse.ST_Transform(g db2gse.ST_Geometry, cr db2gse.coordref)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle TRANSFORM_TEST mit den beiden Linienspalten L1 und L2.

```
CREATE TABLE TRANSFORM_TEST (tid integer, l1 db2gse.ST_LineString,
l2 db2gse.ST_LineString)
```

Die folgende Anweisung INSERT fügt eine Linienfolge mit einer SRID von 102 in l1 ein.

```
INSERT INTO TRANSFORM_TEST VALUES (1, db2gse.ST_LineFromText('linestring
    (10.01 40.43, 92.32 29.89)',
    db2gse.coordref()..srid(102)),NULL)
```

Die Funktion ST_Transform wandelt die Linienfolge von L1 aus dem der SRID 102 zugeordneten Koordinatenbezug in den der SRID 105 zugeordneten Koordinatenbezug um. Die folgende Anweisung UPDATE speichert die umgewandelte Linienfolge in Spalte l2.

```
UPDATE TRANSFORM_TEST SET l2 = db2gse.ST_Transform(l1,
    db2gse.coordref()..srid(105))
```

ST_Union

ST_Union verwendet zwei Geometrieobjekte und gibt ein Geometrieobjekt zurück, das die Vereinigung der Eingabeobjekte darstellt.

Syntax

db2gse.ST_Union(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle SENSITIVE_AREAS, die verschiedene Spalten zur Beschreibung der gefährdeten Institutionen enthält. Die Tabelle SENSITIVE_AREAS enthält auch die Spalte ZONE, in der die Polygongeometrie der Institution gespeichert ist.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

Die folgende Anweisung CREATE TABLE erstellt die Tabelle HAZARDOUS_SITES, in der die Identität der Standorte in den Spalten SITE_ID und NAME gespeichert wird. Der tatsächliche geographische Standort jeder Site wird in der Punktspalte LOCATION gespeichert.

```
CREATE TABLE HAZARDOUS_SITES (site_id integer, name varchar(128),
location db2gse.ST_Point);
```

Die folgende Anweisung SELECT verwendet die Funktion ST_Buffer zum Generieren eines Fünf-Meilen-Puffers um die Standorte der Lagerstätten für gefährlichen Abfall. Die Funktion ST_Union generierte Polygone aus der Vereinigungsmenge der gepufferten Polygone um die Abfallstandorte und der sensiblen Bereiche. Die Funktion ST_Area gibt die Vereinigung der Polygonbereiche zurück.

```
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 verwendet zwei Geometrieobjekte und gibt 1 (TRUE) zurück, wenn das erste Objekt im zweiten vollständig enthalten ist; andernfalls wird 0 (FALSE) zurückgegeben.

Syntax

db2gse.ST_Within(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Rückgabetyp

Integer

Beispiele

Im nachfolgenden Beispiel werden zwei Tabellen erstellt. Die erste Tabelle, BUILDINGFOOTPRINTS, enthält die Gebäudegrundflächen einer Stadt. Die zweite Tabelle, LOTS, enthält die Grundstücke der Stadt. Der Stadtplaner möchte sicherstellen, daß alle Gebäudegrundflächen vollständig auf dem jeweiligen Grundstück liegen.

In beiden Tabellen speichert der Datentyp Multipolygon die Geometrie der Gebäudegrundflächen und der Grundstücke. Der Datenbank-Designer hat für beide Merkmale Multipolygone ausgewählt, da die Grundstücke durch natürliche Merkmale wie beispielsweise Flüsse unterbrochen sein können und da die Gebäudegrundflächen häufig mehrere Gebäude umfassen.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);


CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

Mit der folgenden Anweisung SELECT wählt der Stadtplaner zunächst die Gebäude aus, die nicht nicht vollständig auf einem Grundstück liegen.

```
SELECT building_id
  FROM BUILDINGFOOTPRINTS, LOTS
 WHERE db2gse.ST_Within(footprint,lot) = 0;
```

Die erste Abfrage liefert zwar eine Liste aller BUILDING_IDs, deren Gebäudegrundfläche außerhalb eines Grundstückspolygons liegt; sie ermittelt jedoch nicht, ob den übrigen Gebäuden die richtige lot_id zugeordnet wurde. Diese zweite SELECT-Anweisung führt eine Datenintegritätsprüfung mit der Spalte LOT_ID der Tabelle BUILDINGFOOTPRINTS aus.

```
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 konstruiert einen Wert ST_Geometry anhand der bekannten binären Darstellung. Der SRID-Wert Null wird automatisch verwendet.

Syntax

db2gse.ST_WKBToSQL(WKBGeometry Blob(1M))

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle LOTS mit zwei Spalten: Die Spalte LOT_ID, die jedes Grundstück eindeutig kennzeichnet, und die Multipolygonspalte LOT, die die Geometrie jedes Grundstücks enthält.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

Der folgende C-Codeausschnitt enthält ODBC-Funktionen, die in SQL-Funktionen des DB2 Spatial Extender eingebettet sind; diese Funktionen fügen Daten in die Tabelle LOTS ein.

Die Funktion ST_WKBToSQL wandelt WKB-Darstellungen in DB2 Spatial Extender-Geometrien um. Die gesamte Anweisung INSERT wird in eine Zeichenfolge wkb_sql kopiert. Die Anweisung INSERT enthält Parametermarken zum dynamischen Akzeptieren der LOT_ID-Daten und der LOT-Daten.

```
/* 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 konstruiert einen Wert ST_Geometry anhand der bekannten Textdarstellung. Der SRID-Wert Null wird automatisch verwendet.

Syntax

db2gse.ST_WKTToSQL(geometryTaggedText Varchar(4000))

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle GEOMETRY_TEST mit zwei Spalten: die Spalte GID des Typs Integer, die jede Zeile eindeutig kennzeichnet, und die Spalte G1, in der die Geometrie gespeichert wird.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Die folgenden INSERT-Anweisungen fügen die Daten in die Spalten GID und G1 der Tabelle GEOMETRY_TEST ein. Die Funktion ST_WKTToSQL wandelt die Textdarstellung jeder Geometrie in die entsprechende DB2 Spatial Extender-Exemplarunterklasse um.

```
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 verwendet einen Punkt und gibt seine X-Koordinate zurück.

Syntax

ST_X(p ST_Point)

Rückgabetyp

Double

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle X_TEST mit zwei Spalten: der Spalte GID, die die Zeile eindeutig kennzeichnet, und der Punktspalte PT1.

```
CREATE TABLE X_TEST (gid integer, pt1 db2gse.ST_Point)
```

Die folgenden INSERT-Anweisungen fügen zwei Zeilen ein. Eine der Zeilen ist ein Punkt ohne Z-Koordinate oder Maß. Die andere Spalte hat eine Z-Koordinate und ein Maß.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe listet die Spalte GID und die Double-X-Koordinate der Punkte auf.

```
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 verwendet einen Punkt und gibt seine Y-Koordinate zurück.

Syntax

db2gse.ST_Y(p db2gse.ST_Point)

Rückgabetyp

Double

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle Y_TEST mit zwei Spalten: der Spalte GID, die die Zeile eindeutig kennzeichnet, und der Punktspalte PT1.

```
CREATE TABLE Y_TEST (gid integer, pt1 db2gse.ST_Point)
```

Die INSERT-Anweisungen fügen zwei Zeilen ein. Eine der Zeilen ist ein Punkt ohne Z-Koordinate oder Maß. Die andere Spalte hat eine Z-Koordinate und ein Maß.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe listet die Spalte GID und die Double-Y-Koordinate der Punkte auf.

```
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 verwendet einen Punkt und gibt seine Z-Koordinate zurück.

Syntax

Z(p db2gse.ST_Point)

Rückgabetyp

Double

Beispiele

Die folgende Anweisung CREATE TABLE erstellt die Tabelle Z_TEST mit zwei Spalten: der Spalte GID, die die Zeile eindeutig kennzeichnet, und der Punktspalte PT1.

```
CREATE TABLE Z_TEST (gid integer, pt1 db2gse.ST_Point)
```

Die folgenden INSERT-Anweisungen fügen zwei Zeilen ein. Eine der Zeilen ist ein Punkt ohne Z-Koordinate oder Maß. Die andere Spalte hat eine Z-Koordinate und ein Maß.

```
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)))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnisgruppe listet die Spalte GID und die Double-Z-Koordinate der Punkte auf. Die erste Zeile ist NULL, da der Punkt keine Z-Koordinate hat.

```
SELECT gid, db2gse.Z(pt1) "The Z coordinate" FROM Z_TEST

GID         The Z coordinate
----------- ------------------------
          1                        -
          2   +5.00000000000000E+000

  2 record(s) selected.
```

Kapitel 15. Koordinatensysteme

Dieses Kapitel enthält Referenzinformationen zum räumlichen Bezugssystem (SRS) und zu den Koordinatenwerten zum Interpretieren der räumlichen Daten.

Übersicht über die Koordinatensysteme

Die bekannte Textdarstellung des räumlichen Bezugssystems bietet eine Standard-Textdarstellung für Informationen zum räumlichen Bezugssystem. Die Definition der bekannten Textdarstellung wird entsprechend dem POSC/EPSG-Koordinatensystemmodell modelliert.

Ein räumliches Bezugssystem ist ein geographisches Koordinatensystem (Breitengrad-Längengrad), ein projiziertes Koordinatensystem (X,Y) oder ein geozentrisches (X,Y,Z) Koordinatensystem. Das Koordinatensystem besteht aus mehreren Objekten. Jedes Objekt hat ein Schlüsselwort in Großbuchstaben (z. B. DATUM oder UNIT), gefolgt von den durch Kommas getrennten Definitionsparametern des Objekts in Klammern. Manche Objekte bestehen aus anderen Objekten, so daß das Ergebnis eine verschachtelte Struktur darstellt.

Anmerkung: Implementierungen können statt der eckigen Klammern [ ] auch runde Klammern ( ) verwenden und sollten nach Möglichkeit beide Arten von Klammern lesen können.

Die EBNF-Definition (Extended Backus Naur Form) für die Zeichenfolgendarstellung eines Koordinatensystems mit eckigen Klammern lautet wie folgt (siehe Anmerkung oben zur Verwendung der Klammern):

```
<Koordinatensystem> = <projiziertes KS> |
<geographisches KS> | <geozentrisches KS>
<projiziertes KS> = PROJCS["<name>", <geographisches KS>, <projektion>,
                    {<parameter>,}* <lineare einheit>]
<projektion> =
PROJECTION["<name>"]
<parameter> = PARAMETER["<name>", <wert>]
<wert> = <zahl>
```

Ein Koordinatensystem eines Datensatzes wird durch das Schlüsselwort PROJCS definiert, wenn die Daten in projizierten Koordinaten stehen (über GEOGCS bei geographischen Koordinaten, über GEOCCS bei geozentrischen Koordinaten). Das Schlüsselwort PROJCS wird gefolgt von allen "Stücken", die das projizierte Koordinatensystem definieren. Das erste Stück jedes Objekts ist immer der Name. Mehrere Objekte folgen dem Namen des projizierten Koordinatensystems: das geographische Koordinatensystem, die Kartenprojektion, ein oder mehrere Parameter und die lineare Maßeinheit. Alle projizierten Koordinatensysteme basieren auf einem geographischen Koordinatensystem; dieser Abschnitt beschreibt daher zunächst die Stücke, die für ein projiziertes Koordinatensystem spezifisch sind. UTM zone 10N im Faktum NAD83 ist beispielsweise wie folgt definiert:

```
PROJCS["NAD_1983_UTM_Zone_10N",
<geographic cs>,
PROJECTION["Transverse_Mercator"],
PARAMETER["False_Easting",500000.0],
PARAMETER["False_Northing",0.0],
PARAMETER["Central_Meridian",-123.0],
PARAMETER["Scale_Factor",0.9996],
PARAMETER["Latitude_of_Origin",0.0],
UNIT["Meter",1.0]]
```

Der Name und verschiedene Objekte definieren wiederum das geographische Koordinatensystemobjekt: das Faktum, der primäre Längengrad und die Winkeleinheit des Maßes.

```
<geographisches KS> = GEOGCS["<name>", <faktum>, <primärer längengrad>,
<winkeleinheit>]
<faktum> = DATUM["<name>", <spheroid>]
<spheroid> = SPHEROID["<name>", <halbhauptachse>, <inversionsabflachung>]
<halbhauptachse> = <zahl>
    (die Halbhauptachse wird in Metern gemessen; sie muß > 0 sein.)
<inversionsabflachung> = <zahl>
<primärer längengrad> = PRIMEM["<name>", <längengrad>]
<längengrad> = <zahl>
```

Die Zeichenfolge im geographischen Koordinatensystem für UTM zone 10 in NAD83:

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],
UNIT["Degree",0.0174532925199433]]
```

Das Objekt UNIT kann eine Winkeleinheit oder eine lineare Maßeinheit sein:

```
<winkeleinheit> = <einheit>
<lineare einheit> = <einheit>
<einheit> = UNIT["<name>", <umsetzungsfaktor>]
<umsetzungsfaktor> = <zahl>
```

Der Umsetzungsfaktor gibt die Anzahl der Meter (für eine lineare Einheit) oder die Zahl als Bogenmaß (für eine Winkeleinheit) pro Einheit ein; er muß größer als Null sein.

Die vollständige Zeichenfolgendarstellung für UTM Zone 10N lautet also wie folgt:

```
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]]
```

Ein geozentrisches Koordinatensystem ähnelt einem geographischen Koordinatensystem:

```
<geozentrisches KS> = GEOCCS["<name>", <faktum>, <primärer längengrad>,
<lineare einheit>]
```

Unterstützte lineare Einheiten

Tabelle 57. Unterstützte lineare Einheiten

Einheit	Umsetzungsfaktor
Meter	1,0
Fuß (International)	0,3048
Fuß (US)	12/39,37
Modifizierter amerikanischer Fuß	12,0004584/39,37
Clarke's Fuß	12/39,370432
Indischer Fuß	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 (Indian)	36/39,370141
Yard (Sears)	36/39,370147
Fathom	1,8288
Nautische Meile	1852,0

Unterstützte Winkeleinheiten

Tabelle 58. Unterstützte Winkeleinheiten

Einheit	Umsetzungsfaktor
Bogenmaß	1,0
Dezimalgrad	p/180
Dezimale Minute	(p/180)/60
Dezimale Sekunde	(p/180)/36000
Gon	p/200
Grad	p/200

Unterstützte Spheroide

Tabelle 59. Unterstützte Spheroide

Name	Halbhauptachse	Inversionsabflachung
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 (Namibia)	6377483,865	299,1528128
Clarke 1866	6378206,4	294,9786982
Clarke 1866 (Michigan)	6378693,704	294,978684677
Clarke 1880	6378249,145	293,465
Clarke 1880 (Arc)	6378249,145	293,466307656
Clarke 1880 (Benoit)	6378300,79	293,466234571
Clarke 1880 (IGN)	6378249,2	293,46602
Clarke 1880 (RGS)	6378249,145	293,465
Clarke 1880 (SGA)	6378249,2	293,46598
Everest 1830	6377276,345	300,8017
Everest 1975	6377301,243	300,8017
Everest (Sarawak and Sabah)	6377298,556	300,8017
Modified Everest 1948	6377304,063	300,8017
Fischer 1960	6378166	298,3
Fischer 1968	6378150	298,3
Modified Fischer (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

Tabelle 59. Unterstützte Spheroide (Forts.)

Name	Halbhauptachse	Inversionsabflachung
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
Kugel (Radius = 1.0)	1	0
Kugel (Radius = 6371000 m)	6371000	0
Kugel (Radius =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

Unterstützte geodätische Fakten

Tabelle 60. Unterstützte geodätische Fakten

Adindan	Lissabon
Afgooye	Loma Quintana
Agadez	Lome
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
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)
Elfenbeinküste	Padang 1884
Datum 73	Palästina 1923
Deir ez Zor	Pointe Noire

Tabelle 60. Unterstützte geodätische Fakten (Forts.)

Deutsche Hauptdreiecksnetz	Provisional South American Datum 1956
Douala	Pulkovo 1942
European Datum 1950	Qatar
European Datum 1987	Qatar 1948
Ägypten 1907	Qornoq
European Reference System 1989	RT38
Fahud	Südamerika 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
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

Unterstützte primäre Längengrade

Tabelle 61. Unterstützte primäre Längengrade

Standort	Koordinaten
Greenwich	0° 0' 0"
Bern	7° 26' 22,5" E
Bogota	74° 4' 51,3" W
Brüssel	4° 22' 4,71" E
Ferro	17° 40' 0" W
Jakarta	106° 48' 27,79" E
Lissabon	9° 7' 54,862" W
Madrid	3° 41' 16,58" W
Paris	2° 20' 14,025" E
Rom	12° 27' 8,4" E
Stockholm	18° 3' 29" E

Unterstützte Kartenprojektionen

Tabelle 62. Unterstützte Kartenprojektionen

Zylindrische Projektionen	Pseudozylindrische Projektionen
Behrmann	Craster parabolic
Cassini	Eckert I
Cylindrical equal area	Eckert II
Equirectangular	Eckert III
Gall's stereographic	Eckert IV
Gauss-Kruger	Eckert V
Mercator	Eckert VI
Miller cylindrical	McBryde-Thomas flat polar quartic
Oblique	Mercator (Hotine) Mollweide
Plate-Carée	Robinson
Times	Sinusoidal (Sansom-Flamsteed)
Transverse Mercator	Winkel I

Konische Projektionen

Tabelle 63. Konische Projektionen

Albers conic equal-area	Chamberlin trimetric
Bipolar oblique conformal conic	Two-point equidistant
Bonne	Hammer-Aitoff equal-area
Equidistant conic	Van der Grinten I
Lambert conformal conic	Verschiedene
Polyconic	Alaska series E
Einfache konische	Alaska Grid (Modified-Stereographic by Snyder)

Azimutale oder planare Projektionen

- Azimuthal equidistant
- General vertical near-side perspective
- Gnomonic
- Lambert Azimuthal equal-area
- Orthographic
- Polar-Stereographic
- Stereographic

Kartenprojektionsparameter

Tabelle 64. Kartenprojektionsparameter

Parameter	Beschreibung
central_meridian	Der als Ursprung der X-Koordinaten ausgewählte Längengrad.
scale_factor	Im allgemeinen zur Reduzierung der Verzerrung bei der Kartenprojektion verwendet.
standard_parallel_1	Ein Breitengrad, der normalerweise keine Verzerrung aufweist. Auch für ″maßstabsgerechter Breitengrad″ verwendet.
standard_parallel_2	Ein Breitengrad, der normalerweise keine Verzerrung aufweist.
longitude_of_center	Der Längengrad, der den zentralen Punkt der Kartenprojektion definiert.

Tabelle 64. Kartenprojektionsparameter (Forts.)

Parameter	Beschreibung
latitude_of_center	Der Breitengrad, der den zentralen Punkt der Kartenprojektion definiert.
latitude_of_origin	Der Breitengrad, der als Ursprung der Y-Koordinaten ausgewählt wurde.
false_easting	Wird zu den X-Koordinaten addiert. Wird verwendet, um positive Werte zu erhalten.
false_northing	Wird zu den Y-Koordinaten addiert. Wird verwendet, um positive Werte zu erhalten.
azimuth	Der Winkel östlich von Nord, der die Mittellinie einer schiefen Projektion definiert.
longitude_of_point_1	Der Längengrad des ersten für eine Kartenprojektion erforderlichen Punkts.
latitude_of_point_1	Der Breitengrad des ersten für eine Kartenprojektion erforderlichen Punkts.
longitude_of_point_2	Der Längengrad des zweiten für eine Kartenprojektion erforderlichen Punkts.
latitude_of_point_2	Der Breitengrad des zweiten für eine Kartenprojektion erforderlichen Punkts.
longitude_of_point_3	Der Längengrad des dritten für eine Kartenprojektion erforderlichen Punkts.
latitude_of_point_3	Der Breitengrad des dritten für eine Kartenprojektion erforderlichen Punkts.
landsat_number	Die Nummer eines Landsat-Satelliten.
path_number	Die Umlaufbahnnummer für einen bestimmten Satelliten.
perspective_point_height	Die Höhe des perspektivischen Punkts der Kartenprojektion über der Erde.
fipszone	Zonennummer des State Plane-Koordinatensystems.
zone	UTM-Zonennummer.

Kapitel 16. Dateiformate für räumliche Daten

Dieses Kapitel dokumentiert die bekannten Darstellungen des DB2 Spatial Extender. Die Darstellungen werden als *bekannt* (well-known) bezeichnet, da sie von ESRI bereitgestellt werden und nicht DB2 Spatial Extender-spezifisch sind. Drei Arten räumlicher Werte sind wichtig, um das Importieren und Exportieren räumlicher Daten zu verstehen:

- Die bekannten Textdarstellungen des Open GIS Consortium (OGC)
- Die bekannten Binärdarstellungen des OGC (WKB)
- Die ESRI-Formdarstellungen

Die bekannten OGC-Textdarstellungen

DB2 Spatial Extender hat verschiedene Funktionen, die Geometrien aus Textbeschreibungen generieren:

ST_GeomFromText
: Erstellt eine Geometrie aus einer Textdarstellung eines beliebigen Geometrietyps.

ST_PointFromText
: Erstellt einen Punkt aus einer Textdarstellung eines Punkts.

ST_LineFromText
: Erstellt eine Linienfolge aus der Textdarstellung einer Linienfolge.

ST_PolyFromText
: Erstellt ein Polygon aus der Textdarstellung eines Polygons.

ST_MPointFromText
: Erstellt eine Mehrpunktangabe aus der Textdarstellung einer Mehrpunktangabe.

ST_MLineFromText
: Erstellt eine Mehrlinienfolge aus der Textdarstellung einer Mehrlinienfolge.

ST_MPolyFromText
: Erstellt ein Multipolygon aus der Textdarstellung eines Multipolygons.

Die Textdarstellung ist eine ASCII-Zeichenfolge, die den Austausch der Geometrie in ASCII-Textform ermöglicht. Sie können diese Funktionen in einer Sprache der dritten oder vierten Generation (3GL oder 4GL) verwenden, da sie keine Definition spezieller Programmstrukturen erfordern. Die Funktion ST_AsText setzt eine vorhandene Geometrie um in eine Textdarstellung.

Jeder Geometrie hat eine bekannte Textdarstellung, die zur Konstruktion neuer Exemplare des Typs und zum Umsetzen vorhandener Exemplare in Textform für die alphanumerische Anzeige verwendet werden kann.

Die bekannte Textdarstellung einer Geometrie ist wie folgt definiert: die Notation {}* gibt 0 oder mehr Wiederholungen der Zeichen innerhalb der geschweiften Klammern an; die geschweiften Klammern erscheinen nicht in der Liste der Ausgabezeichen.

```
<Geometrie-Befehlstext> :=
| <Punkt-Befehlstext>
| <Linienfolgen-Befehlstext>
| <Polygon-Befehlstext>
| <Mehrpunktangaben-Befehlstext>
| <Mehrlinienfolgen-Befehlstext>
| <Multipolygon-Befehlstext>

<Punkt-Befehlstext> :=
POINT <Punkttext>

<Linienfolgen-Befehlstext> :=
LINESTRING <Linienfolgentext>

<Polygon-Befehlstext> :=
POLYGON <Polygontext>

<Mehrpunktangaben-Befehlstext> :=
MULTIPOINT <Mehrpunktangabentext>

<Mehrlinienfolgen-Befehlstext> :=
MULTILINESTRING <Mehrlinienfolgentext>

<Multipolygon-Befehlstext> :=
MULTIPOLYGON <Multipolygontext>

<Punkttext> := EMPTY
|    <Punkt>
| Z  <PunktZ>
| M  <PunktM>
| ZM <PunktZM>

<Punkt> :=  <x>  <y>
<x> := Literal mit doppelter Genauigkeit
<y> := Literal mit doppelter Genauigkeit
<PunktZ> :=  <x>  <y>  <z>
<x> := Literal mit doppelter Genauigkeit
<y> := Literal mit doppelter Genauigkeit
<z> := Literal mit doppelter Genauigkeit
<PunktM> :=  <x>  <y>  <m>
<x> := Literal mit doppelter Genauigkeit
<y> := Literal mit doppelter Genauigkeit
<m> := Literal mit doppelter Genauigkeit
<PunktZM> :=  <x>  <y>  <z>  <m>
<x> := Literal mit doppelter Genauigkeit
```

```
<y> := Literal mit doppelter Genauigkeit
<z> := Literal mit doppelter Genauigkeit
<m> := Literal mit doppelter Genauigkeit

<Linienfolgentext> := EMPTY
|    ( <Punkttext >   {,
<Punkttext> }*  )
| Z  ( <PunktZ-Text >  {,  <PunktZ-Text> }*  )
| M  ( <PunktM-Text >  {,  <PunktM-Text> }*  )
| ZM ( <PunktZM-Text > {,  <PunktZM-Text> }*  )

<Polygontext> := EMPTY
| ( <Linienfolgentext > {,<Linienfolgentext > }*  )

<Mehrpunkttext> := EMPTY
| ( <Punkttext >   {,  <Punkttext > }*  )

<Mehrlinienfolgentext> := EMPTY
| ( <Linienfolgentext > {,< Linienfolgentext>}*  )

<Multipolygontext> := EMPTY
| ( < Polygontext > {,  < Polygontext > }*  )
```

Die Basisfunktionssyntax lautet:

```
Funktion (<Textbeschreibung>,<SRID>)
```

Die SRID, die Kennung des räumlichen Bezugssystems, und der Primärschlüssel zu der Tabelle SPATIAL_REFERENCES kennzeichnen das räumliche Bezugssystem der Geometrie, das in der Tabelle SPATIAL_REFERENCES gespeichert ist. Bevor eine Geometrie in eine räumliche Spalte eingefügt wird, muß ihre SRID mit der SRID der räumlichen Spalte übereinstimmen.

Die Textbeschreibung besteht aus drei in einfachen Anführungszeichen eingeschlossenen Basiskomponenten. Beispiel.

```
<'Geometrietyp'> ['Koordinatentyp' ] [''Koordinatenliste']
```

Die Angaben haben folgende Bedeutung:

Geometrietyp
: Eine der folgenden Angaben: Punkt, Linienfolge, Polygon, Mehrpunktangabe, Mehrlinienfolge oder Multipolygon.

Koordinatentyp
: Gibt an, ob die Geometrie Z-Koordinaten oder Maße hat oder nicht. Lassen Sie dieses Argument leer, wenn die Geometrie weder Z-Koordinaten noch Maße hat. Andernfalls stellen Sie den Koordinatentyp auf Z ein für Geometrien mit Z-Koordinaten, auf M für Geometrien mit Maßen oder auf ZM für Geometrien mit beiden Merkmalen.

Koordinatenliste

Definiert die Scheitelpunkte der Geometrie. Koordinatenlisten werden durch Kommas getrennt und in Klammern eingeschlossen. Geometrien mit mehreren Komponenten erfordern Gruppen von Klammern, in die die einzelnen Teile der Komponenten eingeschlossen werden. Wenn die Geometrie leer ist, ersetzt das Schlüsselwort EMPTY die Koordinate.

Tabelle 65 zeigt eine vollständige Liste der Beispiele aller möglichen Textdarstellungen.

Tabelle 65. Geometrietypen und ihre Textdarstellung

Geometrietyp	Textbeschreibung	Kommentar
point	point empty	Leerer Punkt
point	point z empty	Leerer Punkt mit Z-Koordinate
point	point m empty	Leerer Punkt mit Maß
point	point zm empty	Leerer Punkt mit Z-Koordinate und Maß
point	point (10.05 10.28)	Punkt
point	point z (10.05 10.28 2.51)	Punkt mit Z-Koordinate
point	point m (10.05 10.28 4.72)	Punkt mit Maß
point	point zm (10.05 10.28 2.51 4.72)	Punkt mit Z-Koordinate und Maß
linestring	linestring empty	Leere Linienfolge
linestring	linestring z empty	Leere Linienfolge mit Z-Koordinaten
linestring	linestring m empty	Leere Linienfolge mit Maß
linestring	linestring zm empty	Leere Linienfolge mit Z-Koordinaten und Maß
linestring	linestring (10.05 10.28 , 20.95 20.89)	Linienfolge
linestring	linestring z (10.05 10.28 3.09, 20.95 31.98 4.72, 21.98 29.80 3.51)	Linienfolge mit Z-Koordinaten
linestring	linestring m (10.05 10.28 5.84, 20.95 31.98 9.01, 21.98 29.80 12.84)	Linienfolge mit Maß
linestring	linestring zm ( )	Linienfolge mit Z-Koordinaten und Maß
polygon	polygon empty	Leeres Polygon

Tabelle 65. Geometrietypen und ihre Textdarstellung (Forts.)

Geometrietyp	Textbeschreibung	Kommentar
polygon	polygon z empty	Leeres Polygon mit Z-Koordinaten
Polygon	polygon m empty	Leeres Polygon mit Maß
polygon	polygon zm empty	Leeres Polygon mit Z-Koordinaten und Maß
polygon	polygon ((10 10, 10 20, 20 20, 20 15, 10 10))	Polygon
polygon	polygon z (( ))	Polygon mit Z-Koordinaten
polygon	polygon m (( ))	Polygon mit Maß
polygon	polygon zm (( ))	Polygon mit Z-Koordinaten und Maß
multipoint	multipoint empty	Leere Mehrpunktangabe
multipoint	multipoint z empty	Leere Mehrpunktangabe mit Z-Koordinaten
multipoint	multipoint m empty	Leere Mehrpunktangabe mit Maß
multipoint	multipoint zm empty	Leere Mehrpunktangabe mit Z-Koordinaten und Maß
multipoint	multipoint empty	Leere Mehrpunktangabe
multipoint	multipoint (10 10, 20 20)	Mehrpunktangabe mit zwei Punkten
multipoint	multipoint z (10 10 2, 20 20 3)	Mehrpunktangabe mit Z-Koordinaten
multipoint	multipoint m (10 10 4, 20 20 5)	Mehrpunktangabe mit Maß
multipoint	multipoint zm (10 10 2 4, 20 20 3 5)	Mehrpunktangabe mit Z-Koordinaten und Maß
multilinestring	multilinestring empty	Leere Mehrlinienfolge
multilinestring	multilinestring z empty	Leere Mehrlinienfolge mit Z-Koordinaten
multilinestring	multilinestring m empty	Leere Mehrlinienfolge mit Maß
multilinestring	multilinestring zm empty	Leere Mehrlinienfolge mit Z-Koordinaten mit Maß
multilinestring	multilinestring (( ))	Mehrlinienfolge

Tabelle 65. Geometrietypen und ihre Textdarstellung (Forts.)

Geometrietyp	Textbeschreibung	Kommentar
multilinestring	multilinestring z (( ))	Mehrlinienfolge mit Z-Koordinaten
multilinestring	multilinestring m (( ))	Mehrlinienfolge mit Maß
multilinestring	multilinestring zm (( ))	Mehrlinienfolge mit Z-Koordinaten mit Maß
multipolygon	multipolygon empty	Leeres Multipolygon
multipolygon	multipolygon z empty	Leeres Multipolygon mit Z-Koordinaten
multipolygon	multipolygon m empty	Leeres Multipolygon mit Maß
multipolygon	multipolygon z	Leeres Multipolygon mit Z-Koordinaten und Maß
multipolygon	multipolygon ((( )))	Multipolygon
multipolygon	multipolygon z ((( )))	Multipolygon mit Z-Koordinates
multipolygon	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)))	Multipolygon mit Maß
multipolygon	multipolygon zm ((( )))	Multipolygon mit Z-Koordinaten und Maß

Die bekannten OGC-Binärdarstellungen (WKB)

DB2 Spatial Extender hat verschiedene Funktionen, die Geometrien aus Binärdarstellungen generieren:

ST_GeomFromWKB
Erstellt eine Geometrie aus einer bekannten Binärdarstellung eines beliebigen Geometrietyps.

ST_PointFromWKB
Erstellt einen Punkt aus einer bekannten Binärdarstellung eines Punkts.

ST_LineFromWKB
Erstellt eine Linienfolge aus einer bekannten Binärdarstellung einer Linienfolge.

ST_PolyFromWKB
Erstellt ein Polygon aus einer bekannten Binärdarstellung eines Polygons.

ST_MPointFromWKB
Erstellt eine Mehrpunktangabe aus einer bekannten Binärdarstellung einer Mehrpunktangabe.

ST_MLineFromWKB
Erstellt eine Mehrlinienfolge aus einer bekannten Binärdarstellung einer Mehrlinienfolge.

ST_MPolyFromWKB
Erstellt ein Multipolygon aus einer bekannten Binärdarstellung eines Multipolygons.

Die bekannte Binärdarstellung ist ein fortlaufender Byte-Datenstrom. Sie ermöglicht den Austausch der Geometrie zwischen einem ODBC-Client und einer SQL-Datenbank in binärer Form. Da diese Geometriefunktionen die Definition von C-Programmstrukturen für die Zuordnung der Binärdarstellung erfordert, sind diese Funktionen für Sprachen der dritten Generation (3GL) konzipiert. Für Umgebungen mit Sprachen der vierten Generation (4GL) sind diese Funktionen nicht geeignet. Die Funktion ST_AsBinary setzt eine vorhandene Geometrie um in eine Binärdarstellung.

Die bekannte Binärdarstellung für die Geometrie ergibt sich durch die Serialisierung eines Geometrieexemplars als Folge numerischer Typen. Diese Typen werden aus der Gruppe (unsigned integer, double) gezogen, und jeder numerische Typ wird als Folge von Bytes serialisiert. Die Typen werden mit einer der beiden rein definierten binären Standarddarstellungen für numerische Typen (NDR, XDR) serialisiert. Ein den serialisierten Bytes vorangestellter Ein-Byte-Befehl beschreibt die spezifische Binärcodierung (NDR oder XDR) für einen Geometrie-Bytestrom. Der einzige Unterschied zwischen den beiden Codierungen der Geometrie ist die Byteanordnung: Die XDR-Codierung entspricht Big Endian; die NDR-Codierung entspricht Little Endian.

Definition numerischer Typen

Ein *unsigned integer* (ganze Zahl ohne Vorzeichen) ist ein 32-Bit-Datentyp (4 Byte), der eine nicht negative Ganzzahl im Bereich [0, 4294967295] codiert.

Ein *double* (Gleitkommazahl mit doppelter Genauigkeit) ist ein 64-Bit-Datentyp (8 Byte) mit doppelter Genauigkeit, der eine Zahl mit doppelter Genauigkeit mit dem IEEE 754 Double Precision Format codiert.

Diese Definitionen gelten sowohl für XDR als auch für NDR.

XDR-Codierung (Big Endian) numerischer Typen

Die XDR-Darstellung eines unsigned integer ist Big Endian (das erste Byte ist das signifikanteste Byte).

Die XDR-Darstellung eines double ist Big Endian (das erste Bit ist das Vorzeichenbit).

NDR-Codierung (Little Endian) numerischer Typen

Die XDR-Darstellung eines unsigned integer ist Little Endian (das erste Byte ist das am wenigsten signifikante Byte).

Die XDR-Darstellung eines double ist Little Endian (das letzte Bit ist das Vorzeichenbit).

Umsetzung zwischen NDR und XDR

Die Umsetzung zwischen den Datentypen NDR und XDR für unsigned integers und doubles ist eine einfache Operation. Sie umfaßt die Umkehrung der Byteanordnung innerhalb jedes unsigned integer oder double im Bytestrom.

Beschreibung der WKBGeometry-Byteströme

Dieser Abschnitt beschreibt die bekannte Binärdarstellung für die Geometrie. Der Grundbaustein ist der Bytestrom für einen Punkt, der aus zwei doubles besteht. Die Byteströme für andere Geometrien werden mit den Byteströmen für bereits definierte Geometrien erstellt.

```
// Basic Type definitions
// byte : 1 byte
// uint32 : 32 bit unsigned integer  (4 bytes)
// double : double precision number (8 Byte)

// 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;
  }
};
```

Die folgende Abbildung zeigt eine NDR-Darstellung.

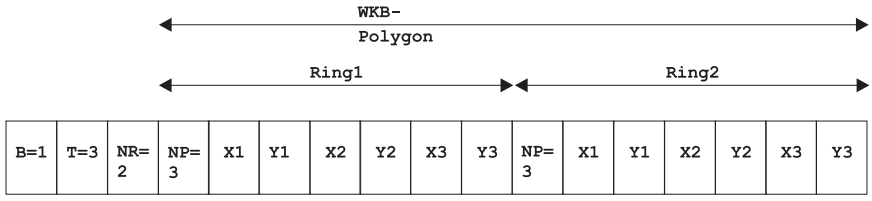


Abbildung 39. Darstellung im NDR-Format. (B=1) des Typs Polygon (T=3) mit 2 linears (NR=2), jeder Ring hat 3 Punkte (NP=3).

Aussagen zur WKB-Darstellung

Die bekannte Binärdarstellung für die Geometrie ist für die Darstellung der Exemplare der Geometrietypen konzipiert; dies wird im Geometrie-Objektmodell und in der OpenGIS Abstract-Spezifikation beschrieben.

Diese Aussagen haben folgende Implikation für Ringe, Polygone und Multipolygone:

Lineare Ringe
Ringe sind einfach und geschlossen; dies bedeutet, daß lineare Ringe sich nicht selbst schneiden können.

Polygone
Keine zwei linearen Ringe innerhalb der Begrenzung eines Polygons können einander schneiden. Die linearen Ringe in der Begrenzung eines Polygons können sich in maximal einem einzigen Punkt berühren.

Multipolygone
Die Innenbereiche von zwei Polygonen, die Elemente eines Multipolygons sind, können sich nicht schneiden. Die Begrenzungen zweier beliebiger Polygone, die Elemente eines Multipolygons sind, können sich nur in einer endlichen Anzahl von Punkten schneiden.

Die ESRI-Formdarstellungen

DB2 Spatial Extender hat verschiedene Funktionen, die Geometrien aus ESRI-Formdarstellungen generieren. Zusätzlich zu den von der bekannten binären GIS-Darstellung unterstützten zweidimensionalen Darstellungen unterstützt die ESRI-Formdarstellung auch auch wahlfreie Z-Koordinaten und Maße. Die folgenden Funktionen generieren Geometrien aus einer ESRI-Form:

ST_GeometryFromShape
: Erstellt eine Geometrie aus einer Formdarstellung eines beliebigen Geometrietyps.

ST_PointFromShape
: Erstellt einen Punkt aus einer Formdarstellung eines Punkts.

ST_LineFromShape
: Erstellt eine Linienfolge aus einer Formdarstellung einer Linienfolge.

ST_PolyFromShape
: Erstellt ein Polygon aus einer Formdarstellung eines Polygons.

ST_MPointFromShape
: Erstellt eine Mehrpunktangabe aus einer Formdarstellung einer Mehrpunktangabe.

ST_MLineFromShape
: Erstellt eine Mehrlinienfolge aus einer Formdarstellung einer Mehrlinienfolge.

ST_MPolyFromShape
: Erstellt ein Multipolygon aus einer Formdarstellung eines Multipolygons.

Die allgemeine Syntax dieser Funktionen ist gleich. Das erste Argument ist die Formdarstellung, die als BLOB-Datentyp (großes Binärobjekt) eingegeben wird. Das zweite Argument ist die Kennung des räumlichen Bezugs, die der Geometrie zugeordnet werden soll. Die Funktion GeometryFromShape hat die folgende Syntax:

```
db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)
```

Da diese Formfunktionen die Definition von C-Programmstrukturen für die Zuordnung der Binärdarstellung erfordern, sind diese Funktionen für Sprachen der dritten Generation (3GL) konzipiert und für Umgebungen der vierten Generation nicht geeignet. Die Funktion AsShape setzt den Geometriewert um in eine ESRI-Formdarstellung.

Der Formtyp 0 kennzeichnet eine Nullform ohne geometrische Daten für die Form.

Wert	Formtyp
0	Nullform (Null Shape)
1	Punkt (Point)
3*	Mehrfachlinie (PolyLine)
5	Polygon
8	Mehrpunktangabe (MultiPoint)
11	PunktZ (PointZ)
13	MehrfachlinieZ (PolyLineZ)
15	PolygonZ (PolygonZ)
18	MehrpunktangabeZ (MultiPointZ)
21	PunktM (PointM)
23	MehrfachlinieM (PolyLineM)
25	PolygonM (PolygonM)
28	MehrpunktangabeM (MultiPointM)

Anmerkung: * Die in dieser Liste nicht angegebenen Formtypen (2, 4, 6 usw.) sind zur späteren Verwendung reserviert.

Formtypen im XY-Raum

Punkt

Ein Punkt besteht aus einem Paar Koordinaten mit doppelter Genauigkeit in der Anordnung X, Y.

Tabelle 66. Punkt-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	1	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	J	J	Double	1	Little

Mehrpunktangabe

Eine Mehrpunktangabe besteht aus einer Gruppe von Punkten. Der Begrenzungsrahmen wird in der Anordnung Xmin, Ymin, Xmax, Ymax gespeichert.

Tabelle 67. Mehrpunkt-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	8	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little

Mehrfachlinie

Eine Mehrfachlinie ist eine geordnete Gruppe von Scheitelpunkten, die aus einem oder mehreren Teilen bestehen. Ein Teil ist eine verbundene Sequenz von zwei oder mehr Punkten. Punkte können miteinander verbunden sein oder auch nicht. Teile können sich schneiden oder auch nicht.

Da diese Spezifikation aufeinanderfolgende Punkte mit identischen Koordinaten zuläßt, müssen diese Fälle beim Lesen der Formdatei berücksichtigt werden. Andererseits sind degenerierte Teile mit einer Länge Null nicht zulässig.

Die Felder für eine Mehrfachlinie sind:

Box Der Begrenzungsrahmen der Mehrfachlinie, gespeichert in der Anordnung Xmin, Ymin, Xmax, Ymax.

NumParts
Die Anzahl der Teile in der Mehrfachlinie.

NumPoints
Die gesamte Anzahl von Punkten für alle Teile.

Parts Eine Feldgruppe von Längen-NumParts. Jede Mehrfachlinie speichert den Index ihres ersten Punkts in der Punktfeldgruppe. Feldgruppenindizes sind auf Null bezogen.

Points Eine Feldgruppe von Längen-NumPoints. Die Punkte für jedes Teil in der Mehrfachlinie werden von Ende zu Ende gespeichert. Die Punkte für Teil 2 folgen den Punkten für Teil 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startpunkts für jedes Teil. Es gibt keine Begrenzer in der Punktfeldgruppe zwischen den Teilen.

Tabelle 68. Mehrfachlinien-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	3	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little

Anmerkung: X = 44 + 4 * NumParts.

Polygon

Ein Polygon besteht aus einem oder mehreren Ringen. Ein Ring ist eine verbundene Folge von vier oder mehr Punkten, die eine geschlossene Schleife bilden, die sich nicht selbst schneidet. Ein Polygon kann mehrere äußere Ringe enthalten. Die Anordnung der Scheitelpunkt oder die Ausrichtung eines Rings gibt an, welche Seite des Rings der Innenbereich des Polygons ist. Der Bereich, der beim Entlanggehen am Ring in Richtung der Scheitelpunkte auf der rechten Seite liegt, ist der Innenbereich des Polygons. Scheitelpunkte von Polygonen, die Löcher in Polygonen definieren, sind gegen den Uhrzeigersinn angeordnet. Scheitelpunkte für einen einzelnen Polygonring sind daher immer im Uhrzeigersinn angeordnet. Die Ringe eines Polygons werden als Teile bezeichnet.

Da diese Spezifikation aufeinanderfolgende Punkte mit identischen Koordinaten zuläßt, müssen diese Fälle beim Lesen der Formdatei berücksichtigt werden. Andererseits sind degenerierte Teile mit einer Länge oder Fläche Null nicht zulässig.

Die Felder für ein Polygon sind:

Box Der Begrenzungsrahmen des Polygons, gespeichert in der Anordnung Xmin, Ymin, Xmax, Ymax.

NumParts
Die Anzahl der Ringe in dem Polygon.

NumPoints
Die gesamte Anzahl von Punkten für alle Ringe.

Parts Eine Feldgruppe von Längen-NumParts. Speichert für jeden Ring den Index seines ersten Punkts in der Punktfeldgruppe. Feldgruppenindizes sind auf Null bezogen.

Points Eine Feldgruppe von Längen-NumPoints. Die Punkte für jeden Ring in dem Polygon werden von Ende zu Ende gespeichert. Die Punkte für Ring 2 folgen den Punkten für Ring 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startpunkts für jeden Ring. Es gibt keine Begrenzer in der Punktfeldgruppe zwischen den Ringen.

Wichtiger Hinweis zu Polygonformen:

- Die Ringe sind geschlossen (der erste und letzte Scheitelpunkt eines Rings MÜSSEN identisch sein).
- Die Reihenfolge der Ringe in der Punktfeldgruppe ist nicht von Bedeutung.
- In einer Formdatei gespeicherte Polygone müssen rein sein. Ein reines Polygon ist wie folgt definiert:
  - Es schneidet sich nicht selbst. Dies bedeutet, daß ein Segment, das zu einem Ring gehört, kein Segment schneidet, das zu einem anderen Ring gehört. Die Ringe eines Polygons können sich an den Schnittpunkten berühren, aber nicht entlang ihrer Segmente. Kolineare Segmente werden als schneidend betrachtet.
  - Die Innenseite des Polygons befindet sich auf der "richtigen" Seite der Linie, durch die es definiert ist. Der Bereich, der beim Entlanggehen am Ring in Richtung der Scheitelpunkte auf der rechten Seite liegt, ist der Innenbereich des Polygons. Scheitelpunkte für einen einzelnen Polygonring sind daher immer im Uhrzeigersinn angeordnet. Ringe, die Löcher in diesen Polygonen definieren, sind gegen den Uhrzeigersinn angeordnet.

  "Nicht reine" Polygone liegen vor, wenn die Ringe, die Löcher in dem Polygon definieren, ebenfalls im Uhrzeigersinn angeordnet sind; dies bewirkt ein Überlappen der Innenbereiche.

Ein Beispielexemplar eines Polygons:

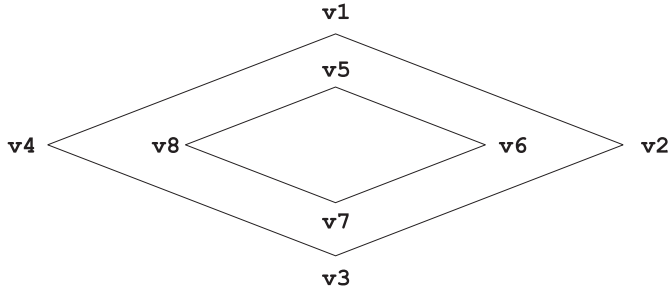


Abbildung 40. Ein Polygon mit einem Loch und acht Scheitelpunkten

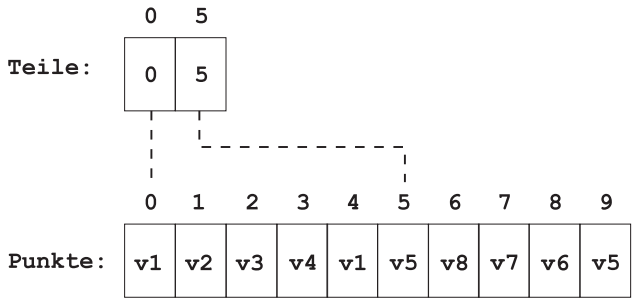


Abbildung 41. Inhalt des Polygon-Bytestroms. NumParts ist 2 und NumPoints ist 10. Beachten Sie, daß die Anordnung der Punkte für die Innenfläche (das Loch) umgekehrt ist.

Tabelle 69. Polygon-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	5	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little

Anmerkung: X = 44 + 4 * NumParts.

Maß-Formtypen in XY-Räumen

PunktM

Ein PunktM besteht aus einem Paar Koordinaten mit doppelter Genauigkeit in der Anordnung X, Y sowie einem Maß M.

Tabelle 70. PunktM-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	21	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	J	J	Double	1	Little
Byte 20	M	M	Double	1	Little

MehrpunktangabeM

Die Felder für eine MehrpunktangabeM sind:

Box Der Begrenzungsrahmen der MehrpunktangabeM, gespeichert in der Anordnung Xmin, Ymin, Xmax, Ymax.

NumPoints
Die Anzahl der Punkte.

Points Eine Feldgruppe von Punkten von Längen-NumPoints.

NumMs
Die Anzahl der folgenden Maße. NumMs kann nur zwei Werte haben: Null, wenn diesem Feld keine Maße folgen, oder gleich NumPoints, wenn Maße vorhanden sind.

M Range
Minimum und Maximum für das Maß für MehrpunktangabeM, gespeichert in der Reihenfolge Mmin, Mmax.

M Array
Eine Feldgruppe von Maßen von Längen-NumPoints.

Tabelle 71. MehrpunktangabeM-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	28	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little
Byte X	NumMs	NumMs	Integer	1	Little
Byte X+4*	Mmin	Mmin	Double	1	Little
Byte X+12*	Mmax	Mmax	Double	1	Little
Byte X+20*	Marray	Marray	Double	NumPoints	Little

Anmerkungen:

1. X = 40 + (16 * NumPoints)
2. * wahlfrei

MehrfachlinieM

Eine Formdatei-MehrfachlinieM besteht aus einem oder mehreren Teilen. Ein Teil ist eine verbundene Sequenz von zwei oder mehr Punkten. Teile können miteinander verbunden sein oder auch nicht. Teile können sich schneiden oder auch nicht.

Die Felder für eine MehrfachlinieM sind:

Box Der Begrenzungsrahmen der MehrfachlinieM, gespeichert in der Anordnung Xmin, Ymin, Xmax, Ymax.

NumParts
Die Anzahl der Teile in der MehrfachlinieM.

NumPoints
Die gesamte Anzahl von Punkten für alle Teile.

Parts Eine Feldgruppe von Längen-NumParts. Speichert für jeden Teil den Index seines ersten Punkts in der Punktfeldgruppe. Feldgruppenindizes sind auf Null bezogen.

Points Eine Feldgruppe von Längen-NumPoints. Die Punkte für jedes Teil in der MehrfachlinieM werden von Ende zu Ende gespeichert. Die Punkte für Teil 2 folgen den Punkten für Teil 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startpunkts für jedes Teil. Es gibt keine Begrenzer in der Punktfeldgruppe zwischen den Teilen.

NumMs
Die Anzahl der folgenden Maße. NumMs kann nur zwei Werte haben: Null, wenn diesem Feld keine Maße folgen, oder gleich NumPoints, wenn Maße vorhanden sind.

M Range
Minimum und Maximum für das Maß für MehrfachlinieM, gespeichert in der Reihenfolge Mmin, Mmax.

M Array
Eine Feldgruppe von Längen-NumPoints. Die Maße für jedes Teil in der MehrfachlinieM werden von Ende zu Ende gespeichert. Die Maße für Teil 2 folgen den Maßen für Teil 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startpunkts für jedes Teil. Es gibt keine Begrenzer in der Maßfeldgruppe zwischen den Teilen.

Tabelle 72. MehrfachlinieM-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	13	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	NumMs	NumMs	Integer	1	Little
Byte Y+4*	Mmin	Mmin	Double	1	Little
Byte Y+12*	Mmax	Mmax	Double	1	Little
Byte Y+20*	Marray	Marray	Double	NumPoints	Little

Anmerkungen:
1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * wahlfrei

PolygonM

Ein PolygonM besteht aus einer Anzahl von Ringen. Ein Ring ist eine geschlossene Schleife, die sich nicht selbst schneidet. Beachten Sie, daß Schnittmengen im XY-Raum berechnet werden, *nicht* im XYM-Raum. Ein PolygonM kann mehrere äußere Ringe enthalten. Die Ringe eines PolygonM werden als Teile bezeichnet.

Die Felder für ein PolygonM sind:

Box Der Begrenzungsrahmen des PolygonM, gespeichert in der Anordnung Xmin, Ymin, Xmax, Ymax.

NumParts
Die Anzahl der Ringe in dem PolygonM.

NumPoints
Die gesamte Anzahl von Punkten für alle Ringe.

Parts Eine Feldgruppe von Längen-NumParts. Speichert für jeden Ring den Index seines ersten Punkts in der Punktfeldgruppe. Feldgruppenindizes sind auf Null bezogen.

Points Eine Feldgruppe von Längen-NumPoints. Die Punkte für jeden Ring in dem PolygonM werden von Ende zu Ende gespeichert. Die Punkte für Ring 2 folgen den Punkten für Ring 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startpunkts für jeden Ring. Es gibt keine Begrenzer in der Punktfeldgruppe zwischen den Ringen.

NumMs

Die Anzahl der folgenden Maße. NumMs kann nur zwei Werte haben: Null, wenn diesem Feld keine Maße folgen, oder gleich NumPoints, wenn Maße vorhanden sind.

M Range

Minimum und Maximum für das Maß für das PolygonM, gespeichert in der Reihenfolge Mmin, Mmax.

M Array

Eine Feldgruppe von Längen-NumPoints. Die Maße für jeden Ring in dem PolygonM werden von Ende zu Ende gespeichert. Die Maße für Ring 2 folgen den Maßen für Ring 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startmaßes für jeden Ring. Es gibt keine Begrenzer in der Maßfeldgruppe zwischen den Ringen.

Wichtige Hinweise zu PolygonM-Formen:

- Die Ringe sind geschlossen (der erste und letzte Scheitelpunkt eines Rings müssen identisch sein).
- Die Reihenfolge der Ringe in der Punktfeldgruppe ist nicht von Bedeutung.

Tabelle 73. PolygonM-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	15	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	NumMs	NumMs	Integer	1	Little
Byte Y+4*	Mmin	Mmin	Double	1	Little
Byte Y+12*	Mmax	Mmax	Double	1	Little
Byte Y+20*	Marray	Marray	Double	NumPoints	Little

Anmerkungen:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * wahlfrei

Formtypen im XYZ-Raum

PunktZ

Ein PunktZ besteht aus einem Triplet Koordinaten mit doppelter Genauigkeit in der Anordnung X, Y, Z sowie einem Maß M.

Tabelle 74. PunktZ-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	11	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	J	J	Double	1	Little
Byte 20	Z	Z	Double	1	Little
Byte 28	Measure	M	Double	1	Little

MehrpunktangabeZ

Ein MehrpunktangabeZ stellt eine PunktZ-Gruppe dar wie folgt:

- Der Begrenzungsrahmen wird in der Anordnung Xmin, Ymin, Xmax, Ymax gespeichert.
- Die Z-Begrenzung wird in der Anordnung Zmin, Zmax gespeichert. Die M-Begrenzung wird in der Anordnung Mmin, Mmax gespeichert.

Tabelle 75. MehrpunktangabeZ-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	18	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little
Byte X	Zmin	Zmin	Double	1	Little
Byte X+8	Zmax	Zmax	Double	1	Little
Byte X+16	Zarray	Zarray	Double	NumPoints	Little
Byte Y	NumMs	NumMs	Integer	1	Little
Byte Y+4*	Mmin	Mmin	Double	1	Little
Byte Y+12*	Mmax	Mmax	Double	1	Little
Byte Y+20*	Marray	Marray	Double	NumPoints	Little

Anmerkungen:

1. X = 40 + (16 * NumPoints); Y = X + 16 + (8 * NumPoints)
2. * wahlfrei

MehrfachlinieZ

Eine MehrfachlinieZ besteht aus einem oder mehreren Teilen. Ein Teil ist eine verbundene Sequenz von zwei oder mehr Punkten. Teile können miteinander verbunden sein oder auch nicht. Teile können sich schneiden oder auch nicht.

Die Felder für eine MehrfachlinieZ sind:

Box Der Begrenzungsrahmen der MehrfachlinieZ, gespeichert in der Anordnung Xmin, Ymin, Xmax, Ymax.

NumParts
Die Anzahl der Teile in der MehrfachlinieZ.

NumPoints
Die gesamte Anzahl von Punkten für alle Teile.

Parts Eine Feldgruppe von Längen-NumParts. Speichert für jeden Teil den Index seines ersten Punkts in der Punktfeldgruppe. Feldgruppenindizes sind auf Null bezogen.

Points Eine Feldgruppe von Längen-NumPoints. Die Punkte für jedes Teil in der MehrfachlinieZ werden von Ende zu Ende gespeichert. Die Punkte für Teil 2 folgen den Punkten für Teil 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startpunkts für jedes Teil. Es gibt keine Begrenzer in der Punktfeldgruppe zwischen den Teilen.

Z Range
Die Minimum- und Maximum-Z-Werte für die MehrfachlinieZ, gespeichert in der Anordnung Zmin, Zmax.

Z Array
Eine Feldgruppe von Längen-NumPoints. Die Z-Werte für jedes Teil in der MehrfachlinieZ werden von Ende zu Ende gespeichert. Die Z-Werte für Teil 2 folgen den Z-Werten für Teil 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startpunkts für jedes Teil. Es gibt keine Begrenzer in der Z-Feldgruppe zwischen den Teilen.

NumMs
Die Anzahl der folgenden Maße. NumMs kann nur zwei Werte haben: Null, wenn diesem Feld keine Maße folgen, oder gleich NumPoints, wenn Maße vorhanden sind.

M Range
Minimum und Maximum für das Maß für die MehrfachlinieZ, gespeichert in der Reihenfolge Mmin, Mmax.

M Array
Eine Feldgruppe von Längen-NumPoints. Die Maße für jedes Teil in der MehrfachlinieZ werden von Ende zu Ende gespeichert. Die Maße für Teil 2 folgen den Maßen für Teil 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startmaßes für jedes Teil. Es gibt keine Begrenzer in der Maßfeldgruppe zwischen den Teilen.

Tabelle 76. MehrfachlinieZ-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	13	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	Zmin	Zmin	Double	1	Little
Byte Y+8	Zmax	Zmax	Double	1	Little
Byte Y+16	Zarray	Zarray	Double	NumPoints	Little
Byte Z	NumMs	NumMs	Integer	1	Little
Byte Z+4*	Mmin	Mmin	Double	1	Little
Byte Z+12*	Mmax	Mmax	Double	1	Little
Byte Z+20*	Marray	Marray	Double	NumPoints	Little

Anmerkungen:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints), Z = Y + 16 + (8 * NumPoints)
2. * wahlfrei

PolygonZ

Ein PolygonZ besteht aus einer Anzahl von Ringen. Ein Ring ist eine geschlossene Schleife, die sich nicht selbst schneidet. Ein PolygonZ kann mehrere äußere Ringe enthalten. Die Ringe eines PolygonZ werden als Teile bezeichnet.

Die Felder für ein PolygonZ sind:

Box Der Begrenzungsrahmen des PolygonZ, gespeichert in der Anordnung Xmin, Ymin, Xmax, Ymax.

NumParts
Die Anzahl der Ringe in dem PolygonZ.

NumPoints
Die gesamte Anzahl von Punkten für alle Ringe.

Parts Eine Feldgruppe von Längen-NumParts. Speichert für jeden Ring den Index seines ersten Punkts in der Punktfeldgruppe. Feldgruppenindizes sind auf Null bezogen.

Points Eine Feldgruppe von Längen-NumPoints. Die Punkte für jeden Ring in dem PolygonZ werden von Ende zu Ende gespeichert. Die Punkte für Ring 2 folgen den Punkten für Ring 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startpunkts für jeden Ring. Es gibt keine Begrenzer in der Punktfeldgruppe zwischen den Ringen.

Z Range
Die Minimum- und Maximum-Z-Werte für den Bogen, gespeichert in der Anordnung Zmin, Zmax.

Z Array
Eine Feldgruppe von Längen-NumPoints. Die Z-Werte für jeden Ring in dem PolygonZ werden von Ende zu Ende gespeichert. Die Z-Werte für Ring 2 folgen den Z-Werten für Ring 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Start-Z-Werts für jeden Ring. Es gibt keine Begrenzer in der Z-Werte-Feldgruppe zwischen den Ringen.

NumMs
Die Anzahl der folgenden Maße. NumMs kann nur zwei Werte haben: Null, wenn diesem Feld keine Maße folgen, oder gleich NumPoints, wenn Maße vorhanden sind.

M Range
Minimum und Maximum für das Maß für das PolygonZ, gespeichert in der Reihenfolge Mmin, Mmax.

M Array
Eine Feldgruppe von Längen-NumPoints. Die Maße für jeden Ring in dem PolygonZ werden von Ende zu Ende gespeichert. Die Maße für Ring 2 folgen den Maßen für Ring 1, usw. Die Teilefeldgruppen enthalten den Feldgruppenindex des Startmaßes für jeden Ring. Es gibt keine Begrenzer in der Maßfeldgruppe zwischen den Ringen.

Wichtige Hinweise zu PolygonZ-Formen:

- Die Ringe sind geschlossen (der erste und letzte Scheitelpunkt eines Rings MÜSSEN identisch sein).
- Die Reihenfolge der Ringe in der Punktfeldgruppe ist nicht von Bedeutung.

Tabelle 77. PolygonZ-Bytestrom, Inhalt

Position	Feld	Wert	Typ	Anzahl	Anordnung
Byte 0	Shape Type	15	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	Zmin	Zmin	Double	1	Little
Byte Y+8	Zmax	Zmax	Double	1	Little
Byte Y+16	Zarray	Zarray	Double	NumPoints	Little
Byte Z	NumMs	NumMs	Integer	1	Little
Byte Z+4*	Mmin	Mmin	Double	1	Little
Byte Z+12*	Mmax	Mmax	Double	1	Little
Byte Z+20*	Marray	Marray	Double	NumPoints	Little

Teil 3. Anhänge und Schlußteil

Bemerkungen

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, daß nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Dienstleistungen in Verbindung mit Fremdprodukten und Fremddienstleistungen liegt beim Kunden, soweit nicht ausdrücklich solche Verbindungen erwähnt sind.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanfragen sind schriftlich an IBM Europe, Director of Licensing, 92066 Paris La Defense Cedex, France, zu richten. Anfragen an obige Adresse müssen auf englisch formuliert werden.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Web-Sites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Web-Sites dar. Das über diese Web-Sites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Web-Sites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne daß eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der Allgemeinen Geschäftsbedingungen der IBM, der Internationalen Nutzungsbedingungen der IBM für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer gesteuerten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Garantie, daß diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Informationen über Produkte anderer Hersteller als IBM wurden von den Herstellern dieser Produkte zur Verfügung gestellt, bzw. aus von ihnen veröffentlichten Ankündigungen oder anderen öffentlich zugänglichen Quellen entnommen. IBM hat diese Produkte nicht getestet und übernimmt im Hinblick auf Produkte anderer Hersteller keine Verantwortung für einwandfreie Funktion, Kompatibilität oder andere Ansprüche. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten der IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele der IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogrammes illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden, Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHT-LIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle konform sind, für die diese Beispielprogramme geschrieben werden. Die in diesem Handbuch aufgeführten Beispiele sollen lediglich der Veranschaulichung und zu keinem anderen Zweck dienen. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (Name Ihrer Firma) (Jahr). Teile des vorliegenden Codes wurden aus Beispielprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. _Jahr/Jahre angeben_. Alle Rechte vorbehalten.

Marken

Folgende Namen sind in gewissen Ländern Marken der International Business Machines Corporation.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
IBM System 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
RS/6000
IBM System /370
SP
SQL/DS
SQL/400
System/370
IBM System /390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

Folgende Namen sind in gewissen Ländern Marken oder eingetragene Marken anderer Unternehmen:

Microsoft, Windows und Windows NT sind Marken oder eingetragene Marken von Microsoft Corporation.

Java und alle auf Java basierenden Marken und Logos sowie Solaris sind in gewissen Ländern Marken von Sun Microsystems, Inc.

Tivoli und NetView sind in gewissen Ländern Marken von Tivoli Systems Inc.

UNIX ist eine eingetragene Marke und wird ausschließlich von der X/Open Company Limited lizenziert.

Andere Namen von Unternehmen, Produkten oder Dienstleistungen können Marken anderer Unternehmen sein.

Index

E

F

G

I

J

K

L

M

N

P

Q

R

S

T

U

W

X

Y

Z

Kontaktaufnahme mit IBM

Bei technischen Problemen lesen Sie bitte die entsprechenden Korrekturmaßnahmen im Handbuch *Troubleshooting Guide* und führen Sie diese aus, bevor Sie sich mit der IBM Kundenunterstützung in Verbindung setzen. Mit Hilfe dieses Handbuchs können Sie Informationen sammeln, die die DB2-Kundenunterstützung zur Fehlerbehebung verwenden kann.

Wenn Sie weitere Informationen benötigen oder eines der DB2 Universal Database-Produkte bestellen möchten, setzen Sie sich mit einem IBM Ansprechpartner in einer lokalen Geschäftsstelle oder einem IBM Software-Vertriebspartner in Verbindung.

Telefonische Unterstützung erhalten Sie unter der folgenden Nummer:

- Unter 0180 3/313 233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.

Produktinformationen

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3/313 233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0180/55 090 können Sie Handbücher telefonisch bestellen.

http://www.ibm.com/software/data/
Auf den DB2-World Wide Web-Seiten erhalten Sie aktuelle DB2-Informationen wie Neuigkeiten, Produktbeschreibungen, Schulungspläne und vieles mehr.

http://www.ibm.com/software/data/db2/library/
Mit **DB2 Product and Service Technical Library** können Sie auf häufig gestellte Fragen, Berichtigungen, Handbücher und aktuelle technische DB2-Informationen zugreifen.

Anmerkung: Diese Informationen stehen möglicherweise nur auf Englisch zur Verfügung.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
Auf der Web-Site für die Bestellung internationaler Veröffentlichungen (International Publications) finden Sie Informationen zum Bestellverfahren.

http://www.ibm.com/education/certify/
Das 'Professional Certification Program' auf der IBM Web-Site stellt Zertifizierungstestinformationen für eine Reihe von IBM Produkten, u. a. auch DB2, zur Verfügung.

ftp.software.ibm.com
Melden Sie sich als *anonymous* an. Im Verzeichnis `/ps/products/db2` finden Sie Demo-Versionen, Berichtigungen, Informationen und Tools zu DB2 und vielen zugehörigen Produkten.

comp.databases.ibm-db2, bit.listserv.db2-l
Über diese Internet-Newsgroups können DB2-Benutzer Ihre Erfahrungen mit den DB2-Produkten austauschen.

Für Compuserve: GO IBMDB2
Geben Sie diesen Befehl ein, um auf IBM DB2 Family Forums zuzugreifen. Alle DB2-Produkte werden über diese Foren unterstützt.

In Anhang A des Handbuchs *IBM Software Support Handbook* finden Sie Informationen dazu, wie Sie sich mit IBM in Verbindung setzen können. Rufen Sie die folgende Web-Seite auf, um auf dieses Dokument zuzugreifen: http://www.ibm.com/support/. Wählen Sie anschließend die Verbindung zum IBM Software Support Handbook am unteren Rand der Seite aus.

Anmerkung: In einigen Ländern sollten sich die IBM Vertragshändler an die innerhalb ihrer Händlerstruktur vorgesehene Unterstützung wenden, nicht an die IBM Unterstützungsfunktion.

Antwort

IBM DB2 Spatial Extender
Benutzer- und Referenzhandbuch
Version 7

IBM Form SC12-2894-00

Anregungen zur Verbesserung und Ergänzung dieser Veröffentlichung nehmen wir gerne entgegen. Bitte informieren Sie uns über Fehler, ungenaue Darstellungen oder andere Mängel.

Zur Klärung technischer Fragen sowie zu Liefermöglichkeiten und Preisen wenden Sie sich bitte entweder an Ihre IBM Geschäftsstelle, Ihren IBM Geschäftspartner oder Ihren Händler.

Unsere Telefonauskunft ″HALLO IBM″ (Telefonnr.: 01803/31 32 33) steht Ihnen ebenfalls zur Klärung allgemeiner Fragen zur Verfügung.

Kommentare:

Danke für Ihre Bemühungen.

Sie können ihre Kommentare betr. dieser Veröffentlichung wie folgt senden:
- Als Brief an die Postanschrift auf der Rückseite dieses Formulars
- Als E-Mail an die folgende Adresse: comment@tcvm.vnet.ibm.com

Name	Adresse
Firma oder Organisation	
Rufnummer	E-Mail-Adresse

Antwort
SC12-2894-00

IBM Deutschland Informationssysteme GmbH
SW NLS Center

70548 Stuttgart

Teilenummer: CT7C0DE

Printed in Ireland

CT7C0DE

SC12-2894-00