IBM® DB2® Spatial Extender

사용자 안내 및 참조서

버전 7

SA30-1045-00

IBM® DB2® Spatial Extender

사용자 안내 및 참조서

버전 7

SA30-1045-00

이 책과 이 책이 지원하는 제품을 사용하기 앞서, 345 페이지의 『주의사항』에 있는 일반 정보를 반드시 읽으십시오.

목차

그림

표

이 책에 관하여

이 책은 두 부분으로 나뉘어 있습니다. 첫번째 부분에는 DB2 Spatial Extender에 대한 개념적인 정보가 들어 있으며, Windows NT 및 AIX 시스템에 DB2 Spatial Extender을 설치, 구성, 관리 및 프로그램하는 방법에 대해 설명합니다. 두번째 부분은 사용자가 DB2 Spatial Extender와 함께 사용하는 저장 프로시듀어, 지오메트리, 함수, 메시지 및 카탈로그 뷰에 관한 참조 정보로 구성됩니다.

이 책의 사용자

이는 공간 환경을 설정하는 관리자와 공간 데이터를 이용하는 응용프로그램을 개발하는 응용프로그램 프로그래머를 위한 책입니다.

규약

이 책에서는 다음과 같은 강조표시 규약을 사용합니다.

굵은꼴 명령 및 그래픽 사용자 인터페이스(GUI) 제어사항(예: 필드 이름, 폴더 이름, 메뉴 선택항목)을 나타냅니다.

단칸꼴
: 코딩 또는 사용자가 입력하는 텍스트의 예를 나타냅니다.

기울임꼴
: 사용자가 값으로 대체해야 할 변수를 나타냅니다. 기울임꼴은 책 제목을 표시하고 단어를 강조하는 데도 사용됩니다.

대문자꼴
: SQL 키워드 및 오브젝트 이름(예: 테이블, 뷰 및 서버)을 나타냅니다.

의견서 제출 방법

고객의 의견을 통해 IBM은 고품질 정보를 제공할 수 있습니다. 이 책 또는 그 밖의 DB2 문서에 관하여 가지고 있는 의견을 제출해 주십시오. 다음과 같은 방법 중 하나를 사용하여 의견서를 보내실 수 있습니다.

- 웹에서 의견서를 송신하십시오. http://www.ibm.com/software/data/rcf의 IBM 데이터 관리 온라인 독자 의견서 양식에 액세스할 수 있습니다.
- 전자 우편을 이용할 경우에는 comments@vnet.ibm.com으로 의견서를 송신하십시오. 제품 이름, 제품의 버전 번호 및 책 이름과 파트 번호(적용 가능한 경우)를 반드시 적으십시오. 특정 텍스트에 관한 의견이 있는 경우, 텍스트의 위치를 적으십시오(예를 들어, 장 및 절의 제목, 테이블 번호, 페이지 번호 또는 도움말 주제 제목).

제1부 DB2 Spatial Extender 사용

제1장 DB2 Spatial Extender 관련 정보

이 장에서는 DB2 Spatial Extender의 목적을 설명하고 그것이 처리하는 데이터에 대해 토론하며 DB2 Spatial Extender를 사용하는 방법 등에 대해 설명합니다. 이 장을 이 책의 나머지 부분에 대한 빠른 지침으로 사용해도 됩니다.

DB2 Spatial Extender 목적

DB2 Spatial Extender를 사용하여 지리 대상물에 대한 공간 정보를 생성하고 분석할 수 있도록 오브젝트, 데이터 및 응용프로그램의 복합체인 *지리 정보 시스템*(GIS)을 작성합니다. *지리 대상물*에는 지표면을 비교하는 오브젝트와 이를 점유하는 오브젝트가 포함됩니다. 이들은 자연 환경(예: 강, 숲, 언덕 및 사막)과 문화적 환경(예: 도시, 주거지역, 사무실 빌딩, 이정표 등) 양쪽으로 구성됩니다.

*공간 정보*에는 다음과 같은 사실이 포함됩니다.

- 주변 환경에 대한 지리 대상물의 위치(예: 병원 및 진료소가 놓여 있는 도시 내의 지점 또는 지역 지진대에 대한 도시 거주지의 근접도).
- 지리 대상물이 서로 연관된 방식(예: 특정 강 체계가 특정 영역 내에 있거나 강 체계의 지류를 통과하는 영역의 특정 다리와 같은 정보).
- 하나 이상의 지리 대상물에 적용하는 치수(예: 사무실 빌딩과 그의 구획선 사이의 거리 또는 조류 보존 주계의 길이).

공간 정보는 그 자신만으로 또는 전형적인 관계형 데이터베이스 시스템(RDBMS) 출력과 결합하여 프로젝트를 설계하고 비지니스 및 규정을 결정하는 데 도움을 줄 수 있습니다. 예를 들어, 도 복지 정책 관리자는 복지 정책의 신청자와 수혜자가 지구에서 서비스하는 영역 내에 실제로 살고 있는지 검증해야 합니다. DB2 Spatial Extender는 서비스 영역의 위치와 신청자 및 수혜자의 주소에서 이 정보를 유도할 수 있습니다.

또는 음식점 체인의 소유자가 가까운 도시에서 영업을 원한다고 가정하십시오. 새로운 음식점을 열 장소를 결정하기 위해 소유자는 다음과 같은 질문에 대한 응답이 필요합니다. 즉, 외형적으로 자신의 음식점에 자주 들르게 될 단골들이 이 도시

의 어느 지역에 집중되어 있는가? 본 주요 도로가 어디에 있는가? 범죄 발생률이 가장 낮은 곳이 어디인가? 경쟁 음식점이 어디에 위치하고 있는가? DB2 Spatial Extender는 이 질문에 응답하기 위해 가시적 화면에 공간 정보를 생성하고 기초가 되는 RDBMS는 레이블과 텍스트를 생성하여 화면을 설명할 수 있습니다.

DB2 Spatial Extender의 사용에 대한 기타 여러 예들이 이 책 특히, 61 페이지의 『제7장 공간 정보 검색 및 분석』, 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』 및 173 페이지의 『제14장 SQL 조회에 대한 공간 함수』에 있습니다.

지리 대상물을 나타내는 데이터

이 절에서는 공간 정보를 확보하기 위해 사용자가 생성, 저장 및 관리하는 데이터 개요를 제공합니다. 다루게 되는 주제는 다음과 같습니다.

- 데이터가 지리 대상물을 나타내는 방법
- 공간 데이터의 특성
- 공간 데이터의 생성 방법

데이터가 지리 대상물을 나타내는 방법

DB2 Spatial Extender에서 지리 대상물은 테이블이나 뷰의 행 또는 그와 같은 행의 일부분으로 표현될 수 있습니다. 예를 들어, 3 페이지의 『DB2 Spatial Extender 목적』에서 언급되는 두 가지의 지리 대상물 즉, 사무실 빌딩 및 거주 지역을 고려해 봅시다. 5 페이지의 그림1에서 BRANCHES 테이블의 각 행은 은행의 지점 사무실을 나타냅니다. 5 페이지의 그림1에서 CUSTOMERS 테이블의 각 행은 총괄적으로 은행의 고객을 나타냅니다. 그러나, 각 행의 일부분 특히, 고객의 주소가 들어 있는 셀은 고객의 주소를 나타내는 데 사용될 수 있습니다.

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

그림 1. 지리 대상물을 나타내는 테이블 행; 주소 데이터가 지리 대상물을 나타내는 테이블 행. BRANCHES 테이블의 데이터 행은 은행의 지점 사무실을 나타냅니다. CUSTOMERS 테이블의 주소 데이터에 대한 셀은 고객의 주소를 나타냅니다. 양쪽 테이블의 이름 및 주소는 모두 가상입니다.

그림1의 테이블에는 은행의 지점 및 고객을 나타내고 서술하는 데이터가 들어 있습니다. 그러한 데이터를 *속성 데이터*라고 합니다.

속성 데이터의 부분 집합(지점 및 고객의 주소를 나타내는 값)은 공간 정보를 낳는 값으로 변환될 수 있습니다. 예를 들어, 그림1에 표시된대로 한 지점 사무실의 주소가 92467 Airzone Blvd., San Jose CA 95141입니다. 고객의 주소는 9 Concourt Circle, San Jose CA 95141입니다. DB2 Spatial Extender는 이 주소를 환경면에서 지점 및 고객의 홈에 적합한 장소를 나타내는 값으로 변환할 수 있습니다. 6 페이지의 그림2에서는 그러한 값을 포함하도록 지정된 새 컬럼이 있는 BRANCHES 및 CUSTOMERS 테이블을 보여 줍니다.

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

그림 2. 공간 컬럼이 추가된 테이블. 각 테이블에의 LOCATION 컬럼에는 주소에 해당되는 좌표가 들어 있습니다.

주소 및 유사한 식별자들이 공간 정보에 대한 시작점으로 사용될 때, 이 파생된 값을 *소스 데이터*라고 합니다. 여기에서 파생된 값이 공간 정보를 생성하므로, 이 파생된 값을 *공간 데이터*라고 합니다. 다음 절에서는 공간 데이터에 대해 설명하고 그 관련 데이터 유형을 소개합니다.

공간 데이터 특성

수많은 공간 데이터는 좌표로 구성됩니다. *좌표*는 참조 점에 상대적인 위치를 나타내는 숫자입니다. 예를 들어, 위도는 적도에 상대적인 위치를 나타내는 좌표입니다. 경도는 그리니치 자오선에 상대적인 위치를 나타내는 좌표입니다. 그러므로, Yellowstone National Park의 위치는 위도(적도에서 44.45도 북쪽)와 경도(그리니치 자오선에서 110.40도 서쪽)로 정의됩니다.

위도, 경도, 참조 점 및 기타 관련 매개변수들을 모두 모아 *좌표 시스템*이라고 합니다. 위도와 경도가 아닌 다른 값에 기초한 좌표 시스템도 존재합니다. 이 좌표 시스템에는 그 자신 소유의 위치 치수, 참조 점 및 추가 구별 매개변수가 있습니다.

가장 단순한 공간 데이터 항목은 단일 지리 대상물의 위치를 정의하는 두 좌표로 구성됩니다(*데이터 항목*은 관계형 테이블의 셀을 점유하는 값(들)입니다). 보다 광범위한 공간 데이터 항목은 도로 또는 강과 같은 선형 경로를 정의하는 여러 좌표로 구성됩니다. 세 번째 종류는 영역의 주계를 정의하는 좌표로 구성됩니다. 예를 들어, 토지 구획 또는 범람 지역의 가장자리가 해당됩니다. 이 공간 데이터 항목과

DB2 Spatial Extender가 지원하는 다른 종류의 공간 데이터 항목들은 133 페이지의 『제13장 기하학 및 관련 공간 함수』에서 보다 자세히 설명합니다.

각 공간 데이터 항목은 공간 데이터 유형의 인스턴스입니다. 위치를 마크하는 두 좌표에 대한 데이터 유형은 ST_Point이고, 선형 경로를 정의하는 좌표에 대한 데이터 유형은 ST_LineString이며 주계를 정의하는 좌표에 대한 데이터 유형은 ST_Polygon입니다. 이 유형들은 공간 데이터에 대한 다른 데이터 유형과 함께 단일 계층 구조에 속하는 구조화된 유형들입니다. 39 페이지의 『공간 데이터 유형 관련 정보』에서 계층 구조에 대한 개요를 참조하십시오.

공간 데이터의 발원지

사용자는 다음을 수행하여 공간 데이터를 확보할 수 있습니다.

- 속성 데이터로부터 유도
- 다른 공간 데이터로부터 유도
- 가져오기

속성 데이터를 소스 데이터로 사용

DB2 Spatial Extender는 주소와 같은 속성 데이터로부터 공간 데이터를 유도할 수 있습니다(4 페이지의 『데이터가 지리 대상물을 나타내는 방법』에 언급된대로). 이러한 프로세스를 *지오코딩*이라고 합니다. 포함된 순차를 알려면 6 페이지의 그림2를 『이전』 그림으로 8 페이지의 그림3을 『이후』 그림으로 간주하십시오. 6 페이지의 그림2에서는 BRANCHES 테이블과 CUSTOMERS 테이블 모두 공간 데이터용으로 지정된 빈 컬럼을 가지고 있음을 보여 줍니다. DB2 Spatial Extender가 이 테이블의 주소를 지오코딩하여 주소에 해당하는 좌표를 확보한 후에 그 좌표를 컬럼에 위치시킨다고 가정하십시오. 8 페이지의 그림3에서 이 결과에 대해 설명합니다.

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

그림 3. 소스 데이터에서 파생된 공간 데이터를 포함하는 테이블. CUSTOMERS 테이블의 LOCATION 컬럼에는 지오코더가 ADDRESS, CITY, STATE 및 ZIP 컬럼으로부터 유도한 좌표가 들어 있습니다. 마찬가지로, BRANCHES 테이블의 LOCATION 컬럼에는 지오코더가 테이블의 ADDRESS, CITY, STATE 및 ZIP 컬럼에 있는 주소로부터 유도한 좌표가 들어 있습니다. 이 예는 가상의 것이므로 실제 좌표가 아닌 가상의 좌표가 표시됩니다.

DB2 Spatial Extender는 *지오코더*라는 함수를 사용하여 속성 데이터를 공간 데이터로 변환하고 이 공간 데이터를 테이블 컬럼에 위치시킵니다. 47 페이지의 『지오코딩 관련 정보』에서 지오코더에 대한 자세한 내용을 참조하십시오.

다른 공간 데이터를 소스 데이터로 사용

공간 데이터는 속성 데이터 뿐만 아니라 다른 공간 데이터로부터 생성될 수 있습니다. 예를 들어, 지점들이 BRANCHES 테이블에 정의되어 있는 은행에서는 각 지점의 5마일 이내에 얼마나 많은 고객들이 있는지 알고자 합니다. 은행이 데이터베이스에서 이 정보를 얻기 전에, 각 지점 주위의 5마일 반경 이내에 있는 구역의 정의와 함께 데이터베이스를 제공해야 합니다. DB2 Spatial Extender 함수 ST_Buffer는 그와 같은 정의를 작성할 수 있습니다. 각 지점의 좌표를 입력으로 사용하여 ST_Buffer는 원하는 구역의 주계를 구별하는 좌표를 생성할 수 있습니다. 9 페이지의 그림4에서는 ST_Buffer가 제공하는 정보가 있는 BRANCHES 테이블을 보여 줍니다.

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

그림 4. 기존 공간 데이터로부터 파생된 새로운 공간 데이터가 들어 있는 테이블.
SALES_AREA 컬럼 좌표는 LOCATION 컬럼 좌표로부터 ST_Buffer 함수에 의해 파생됩니다. LOCATION 컬럼의 좌표와 같이 SALES_AREA 컬럼의 좌표도 가상의 것이며 실제 데이터가 아닙니다.

ST_Buffer 외에도 DB2 Spatial Extender는 기존 공간 데이터로부터 새로운 공간 데이터를 유도하는 여러 가지 다른 함수를 제공합니다. 162 페이지의 『기존 기하학에서 새 기하학을 생성하는 함수』에서 ST_Buffer 및 다른 함수에 대한 내용을 참조하십시오.

공간 데이터 가져오기

공간 데이터를 얻기 위한 세 번째 방법은 DB2 Spatial Extender가 지원하는 형식의 파일로부터 공간 데이터를 가져오기하는 것입니다. 317 페이지의 『제16장 공간 데이터에 대한 파일 형식』에서 이 형식에 대한 내용을 참조하십시오. 이 파일에는 일반적으로 지도에 적용되는 데이터 즉, 인구 조사 트랙, 평지, 지진 단층 등이 포함됩니다. 사용자가 생성한 공간 데이터와 결합한 데이터를 사용하여 사용 가능한 지리 정보를 증대시킬 수 있습니다. 예를 들어, 공공 토목 공사 부서에서 주택 지역이 얼마나 많은 위험에 노출되어 있는지 판별해야 할 경우에는, ST_Buffer를 사용하여 그 지역 주변의 구역을 정의할 수 있습니다. 그런 다음, 공공 토목 공사 부서는 평지 및 지진 단층에 대한 데이터를 가져오기하여 문제가 되는 이 영역의 어느 부분이 그 구역과 겹치는지 알 수 있습니다.

DB2 Spatial Extender GIS의 작성 및 사용 방법

DB2 Spatial Extender와 그의 기초가 되는 DB2 RDBMS의 결합된 환경 내에서 DB2 Spatial Extender를 설정하고 GIS 프로젝트를 개발하여 DB2 Spatial Extender GIS를 작성합니다. 사용자는 이 프로젝트를 구현하여 즉, 프로젝트에서 제공하는 정보(공간 및 전형적인 데이터 정보 둘 다)를 생성하고 분석하여 GIS를 사용합니다. 전체적인 과정에는 여러 타스크 세트의 수행 과정이 포함됩니다. 이 절

에서는 이 타스크를 수행할 수 있는 인터페이스에 대해 소개하고 타스크 개념을 제공하며 이를 설명할 수 있도록 시나리오도 제공합니다.

DB2 Spatial Extender 및 관련 기능성에 대한 인터페이스

이 절에서는 DB2 Spatial Extender GIS를 작성하고(즉, GIS에 대한 자원을 설정하고 공간 데이터를 확보하는 등) 이를 사용하는 데(즉, 지리 대상물에 관한 정보를 생성하고 분석함) 쓰는 인터페이스를 살펴봅니다.

다음을 수행하여 DB2 Spatial Extender GIS를 작성할 수 있습니다.

- DB2 제어 센터의 DB2 Spatial Extender 창과 메뉴 선택항목을 사용함. 자세한 내용은 다음을 참조하십시오.
  - 27 페이지의『제3장 자원 설정』
  - 39 페이지의『제4장 공간 컬럼 정의, 공간 컬럼을 계층으로 등록 및 지오코더를 사용하여 공간 컬럼을 유지보수』
  - 47 페이지의『제5장 공간 컬럼 데이터 상주』
  - 59 페이지의『제6장 공간 색인 작성』
- DB2 Spatial Extender 저장 프로시듀어를 호출하는 응용프로그램을 실행함. 65 페이지의『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 그러한 프로그램의 개발에 대한 내용을 참조하십시오.
- 제어 센터 및 응용프로그램 모두를 사용함. 예를 들어, 제어 센터를 사용하여 기본 지오코더를 호출할 수 있습니다. 또한, 다른 지오코더를 사용하려면 응용프로그램에 있는 db2gse.gse_register_gc 저장 프로시듀어를 호출하여 이를 DB2 Spatial Extender에 먼저 등록해야 합니다.47 페이지의『지오코딩 관련 정보』에서 기본값 이외의 지오코더에 대한 자세한 내용을 참조하십시오. 97 페이지의『db2gse.gse_register_gc』에서 db2gse.gse_register_gc 저장 프로시듀어에 대한 자세한 내용을 참조하십시오.
- 제어 센터, 응용프로그램 또는 양쪽 모두를 다른 인터페이스와 조합해서 사용함. 예를 들어, 지오코더와 같은 공간 함수가 생성한 데이터를 보유하는 테이블을 작성하기 위해, 명령행 프로세서 또는 제어 센터 인터페이스를 사용할 수 있습니다.

다음을 수행하여 DB2 Spatial Extender GIS를 사용할 수 있습니다.

- geobrowser를 통해 정보를 그래픽적으로 표현함. geobrowser로는 ESRI(Environmental Systems Research Institute)에서 제공하는 ArcExplorer가 있습니다.
- DB2 제어 센터 또는 명령행 프로세서로부터 명시적으로 SQL 조회를 제출함.
- 응용프로그램으로부터 SQL 조회를 제출함.

DB2 Spatial Extender GIS를 작성 및 사용하기 위해 수행하는 타스크

이 절에서는 DB2 Spatial Extender GIS를 작성하고 사용하기 위한 타스크의 개요를 설명합니다. GIS를 작성하기 위한 타스크에는 DB2 Spatial Extender 설정 및 GIS 프로젝트 개발 과정이 포함됩니다. GIS를 사용하기 위한 타스크에는 프로젝트 구현 과정이 포함됩니다. 이 개요는 DB2 Spatial Extender 설정에서 시작하여 GIS 프로젝트의 개발 및 구현에 대해 다룹니다. 개요에서 다룬 타스크들이 실제로는 어떻게 다양하게 적용되는지를 나타내면서 이 절의 결론을 내립니다.

DB2 Spatial Extender 설정

DB2 Spatial Extender을 설정하려면 다음을 수행하십시오.

1. 플랜을 세워 준비하십시오(개발할 GIS 프로젝트 결정, DB2 Spatial Extender에 사용할 데이터베이스 결정, DB2 Spatial Extender를 관리하고 프로젝트를 개발할 담당자 선택 등).
2. DB2 Spatial Extender를 설치하십시오.
3. GIS 프로젝트를 지원할 장소에 자원을 놓으십시오. 예를 들면, 다음과 같습니다.

   DB2 Spatial Extender가 제공하는 자원

   여기에는 시스템 카탈로그, 공간 데이터 유형, 공간 함수(기본 지오코더 포함) 등이 포함됩니다. 이 자원을 설정하는 타스크를 *공간 데이터 조작에 데이터베이스 사용*이라고 합니다.

   사용자, 벤더 또는 양쪽에서 개발한 지오코더

   기본 지오코더는 미국 주소를 공간 데이터로 변환합니다. 사용자의 소속 및 다른 벤더에서 외부 주소와 다른 종류의 속성 데이터를 공간 데이터로 변환하는 지오코더를 제공할 수 있습니다.

19 페이지의 『제2장 DB2 Spatial Extender 설치』에서 DB2 Spatial Extender 설치에 대한 자세한 내용을 참조하십시오. 제어 센터를 사용하여 자원을 제 자리에 위치시키는 내용에 대해서는 27 페이지의 『제3장 자원 설정』을 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이러한 용도로 응용프로그램을 사용하기 위한 내용을 참조하십시오. 14 페이지의 『공간 및 전형적인 데이터를 통합할 시스템』에서 DB2 Spatial Extender를 설정하기 위한 전반적인 과정을 설명하는 시나리오 내용을 참조하십시오.

GIS 프로젝트 개발 및 구현

GIS 프로젝트를 개발하고 구현하려면 다음을 수행하십시오.

1. 플랜을 세워 준비하십시오(프로젝트에 대한 목표 설정, 필요한 테이블 및 데이터 결정, 사용할 좌표 시스템(들)을 결정 등).
2. 사용할 공간 참조 시스템(들)을 결정하십시오. 일반적으로 좌표값에는 정수, 음수 및 십진수가 포함됩니다. 그러나, DB2 Spatial Extender는 모든 좌표값을 양의 정수 양식으로 저장해야 합니다. *공간 참조 시스템*이란 특정 좌표 시스템의 음수 및 십진수를 DB2 Spatial Extender가 저장할 수 있는 양의 정수로 변환하는 방법을 정의하는 매개변수 세트입니다. 공간 컬럼에 사용할 좌표 시스템을 결정한 후에, 해당 컬럼에 대해 필요한 변환이 발생할 수 있는 공간 참조 시스템을 지정해야 합니다. 기존 공간 참조 시스템이 사용자의 요구에 맞으면 그 시스템을 사용하고 그렇지 않으면 새로 작성할 수 있습니다.
3. 공간 데이터를 포함하는 하나 이상의 컬럼을 정의하여 그를 DB2 Spatial Extender에 등록한 후 지오코더를 사용하여 그를 자동으로 관리하게 하십시오.

   공간 컬럼의 등록에는 DB2 Spatial Extender 카탈로그에 공간 컬럼을 기록하는 과정도 포함됩니다. 공간 컬럼을 등록한 시점부터는 이를 *계층*이라고 하는데, 이는 공간 컬럼에서 생성된 정보가 GIS가 사용자용으로 작성하는 가상적인 지리 도시계획 사업에 지층 또는 계층을 추가하기 때문입니다. 공간 컬럼을 등록한 뒤에는 이에 대해 공간 데이터 조작을 수행할 수 있습니다. 예를 들어, 공간 컬럼을 데이터 상주시키고 그에 대해 공간 색인을 정의할 수 있습니다.
4. 공간 컬럼을 데이터 상주시키려면 다음을 수행하십시오.
   - 지오코더를 요구하는 프로젝트의 경우에는 지오코더에 대한 매개변수를 설정하십시오. 그런 다음, 단일 조작으로 사용 가능한 모든 소스 데이터를 지오코딩한 후 그 결과 좌표를 계층에 로드할 수 있도록 이를 수행하십시오.

- 공간 데이터를 가져오기해야 하는 프로젝트의 경우에는 데이터를 가져오기 하십시오.

5. 공간 컬럼에 대한 액세스를 촉진시키십시오. 특히, 이에는 DB2가 공간 데이터에 빠르게 액세스할 수 있도록 색인을 정의하는 과정과 사용자가 상관 데이터를 효율적으로 검색할 수 있도록 뷰를 정의하는 과정도 포함됩니다. 그와 같은 뷰를 정의한 후에 공간 컬럼을 계층으로 등록해야 합니다.
6. 공간 정보와 관련 비지니스 정보를 생성 및 분석하십시오. 이에는 공간 컬럼과 관련 속성 컬럼을 조회하는 과정도 포함됩니다. 그와 같은 조회에는, 두 지리 대상물 사이의 최소 거리 또는 지리 대상물을 둘러싼 영역을 정의하는 좌표와 같이 다양한 정보를 리턴하는 DB2 Spatial Extender 함수가 포함될 수 있습니다. 8 페이지의 『다른 공간 데이터를 소스 데이터로 사용』 및 205 페이지의 『ST_Buffer』에서 그러한 좌표를 리턴하는 함수 ST_Buffer에 대한 자세한 내용을 참조하십시오. 61 페이지의 『제7장 공간 정보 검색 및 분석』 및 173 페이지의 『제14장 SQL 조회에 대한 공간 함수』에서 공간 함수를 사용하는 조회의 예를 참조하십시오.

제어 센터를 사용하여 GIS 프로젝트를 개발하는 데 포함된 타스크를 수행하는 내용에 대해서는 다음을 참조하십시오.

- 27 페이지의 『제3장 자원 설정』
- 39 페이지의 『제4장 공간 컬럼 정의, 공간 컬럼을 계층으로 등록 및 지오코더를 사용하여 공간 컬럼을 유지보수』
- 47 페이지의 『제5장 공간 컬럼 데이터 상주』
- 59 페이지의 『제6장 공간 색인 작성』

61 페이지의 『제7장 공간 정보 검색 및 분석』에서 제어 센터를 사용하여 GIS 프로젝트를 구현하는 내용을 참조하십시오.

65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 응용프로그램을 사용하여 GIS 프로젝트를 개발하고 구현하는 내용을 참조하십시오.

15 페이지의 『사무소를 확립하고 보험료를 조정하는 프로젝트』에서 전반적인 과정을 설명하는 시나리오를 참조하십시오.

타스크 세트의 다양성

DB2 Spatial Extender GIS를 작성하고 사용하기 위해 수행하는 타스크 세트는 사용자의 요구사항과 사용하는 인터페이스에 따라 내용과 순차가 다양할 수 있습니다. 예를 들어, 공간 데이터를 포함할 컬럼을 정의하여 그를 계층으로 등록한 후 지오코더를 사용하여 그를 자동으로 관리하는 타스크를 고려해 보십시오. 제어 센터를 사용하여 이 타스크들을 하나의 창에서 함께 수행할 수 있습니다. 그러나, 프로그램에서 저장 프로시듀어를 호출할 경우에는 이 타스크들을 개별적으로 수행하고 그 수행 시간을 사용자 재량으로 지정할 수 있습니다.

시나리오: 보험 회사에서 GIS를 갱신하는 경우

이 절에는 앞에서 서술된 타스크 세트를 설명하는 시나리오가 있습니다.

Safe Harbor Real Estate Insurance Company의 정보 시스템 환경에는 DB2 Universal Database 시스템과 개별적인 GIS 데이터베이스 관리 시스템이 포함됩니다. extent 내에서 조회에서는 두 시스템으로부터 데이터 조합을 검색할 수 있습니다. 예를 들어, DB2 테이블은 수입에 관한 정보를 저장하고 GIS 테이블은 회사의 지점 사무실 위치를 저장합니다. 그러므로, 지정된 양의 수입을 올리는 사무실 위치를 찾는 것이 가능합니다. 그러나, 두 시스템의 데이터를 통합할 수는 없으며(예를 들어, 사용자가 DB2 컬럼과 GIS 컬럼을 결합할 수 없음) 조회 최적화와 같은 DB2 서비스를 GIS에 사용할 수 없습니다. 이러한 단점을 없애기 위해 Safe Harbor에서는 DB2 Spatial Extender를 채택하여 새로운 GIS 개발부를 설립합니다. 다음 절에서는 그 부서가 DB2 Spatial Extender를 설정하고 그의 첫번째 프로젝트를 실행하는 방법에 대해 설명합니다.

공간 및 전형적인 데이터를 통합할 시스템

DB2 Spatial Extender를 설정하기 위해 Safe Harbor의 GIS 개발부에서는 다음과 같은 작업을 진행합니다.

1. 부서에서 그의 DB2 환경에 DB2 Spatial Extender를 포함시키려고 준비합니다. 예를 들면 다음과 같습니다.
   a. 부서의 관리팀은 공간 관리팀에게 DB2 Spatial Extender를 설치 및 구현하도록 지시하고 공간 분석팀에게 공간 정보를 생성 및 분석하도록 지시합니다.

b. Safe Harbor의 영업 방침은 고객의 요구사항에 우선적으로 맞추는 것이기 때문에, 관리팀은 고객 관련 정보가 들어 있는 데이터베이스에 DB2 Spatial Extender를 설치하기로 결정합니다. 이 정보의 대부분은 CUSTOMERS라는 테이블에 저장됩니다.

   선택된 데이터베이스를 언급하는 가장 편리한 방법으로서 GIS 개발부 구성원들은 이를 *GIS 데이터베이스*라고 부릅니다. 그러나, 이들은 이 데이터베이스가 GIS 프로젝트 전용으로만 예약될 수 없고 비공간 응용프로그램에서도 이를 여전처럼 계속해서 사용할 수 있음을 알고 있습니다.

2. 공간 관리팀이 DB2 Spatial Extender를 설치합니다.
3. 공간 관리팀이 GIS 프로젝트에 필요한 자원을 설정합니다.
   - 팀에서는 제어 센터를 사용하여 공간 데이터 조작에 GIS 데이터베이스를 사용할 수 있도록 자원을 공급합니다. 이 자원에 DB2 Spatial Extender 카탈로그, 공간 데이터 유형, 공간 함수 등이 포함됩니다.
   - Safe Harbor는 비지니스를 캐나다로 확장하려 하기 때문에, 공간 관리팀은 캐나다 주소를 공간 데이터로 변환하는 지오코더를 캐나다 벤더에게 권유하기 시작합니다.

사무소를 확립하고 보험료를 조정하는 프로젝트

DB2 Spatial Extender 하에서 그의 첫번째 GIS 프로젝트를 실행하기 위해 GIS 개발부에서는 다음과 같이 진행합니다.

1. 부서에서 프로젝트를 개발하려고 준비합니다. 예를 들어,
   - 관리팀은 프로젝트에 대한 목표를 설정합니다.
     - 새로운 지점 사무소를 구축할 장소를 결정함.
     - 위험 지역(교통 사고, 범죄, 홍수, 지진 등의 발생률이 높은 지역)에 대한 고객의 근접 기준에 맞게 보험료를 조정함.
   - GIS 프로젝트는 미국 내의 고객 및 사무소와 관련됩니다. 그러므로, 공간 관리팀은 다음을 결정합니다.
     - Safe Harbor가 영업하는 미국 일부 지역의 위치를 정확하게 정의하는 좌표 시스템을 사용할 것인지.
     - 기본 지오코더가 미국 주소를 지오코딩하기 위해 설계되었기 때문에 이 기본 지오코더를 사용할 것인지.

- 공간 관리팀은 프로젝트의 목표를 충족시키기 위해 어떤 데이터가 필요한지 그리고 이 데이터가 어느 테이블에 들어 있는지를 판별합니다.

2. 제어 센터를 사용하여 공간 관리팀은 두 개의 공간 참조 시스템을 작성합니다. 하나는 사무소 위치를 정의하는 좌표를 DB2 Spatial Extender가 저장할 수 있는 데이터 항목으로 변환하는 방법을 결정합니다. 다른 하나는 고객의 주소를 정의하는 좌표를 DB2 Spatial Extender가 저장할 수 있는 데이터 항목으로 변환하는 방법을 결정합니다.
3. 제어 센터를 사용하여, 공간 관리팀은 공간 데이터를 포함할 컬럼을 정의하고 그를 계층으로 등록한 후 지오코더를 사용하여 그를 자동으로 관리합니다.
   - 팀에서는 LOCATION 컬럼을 CUSTOMERS 테이블에 추가합니다. 테이블에는 고객 주소가 이미 들어 있습니다. 기본 지오코더는 고객 주소를 공간 데이터로 변환하여 이 데이터를 LOCATION 컬럼에 로드합니다.
   - 팀에서는 이제 별도의 GIS에 저장된 데이터가 들어 있는 OFFICES 테이블을 작성합니다. 이 데이터에는 Safe Harbor의 지점 사무소 주소, 지오코더에 의해 이 주소로부터 파생된 공간 데이터 및 각 사무소 주변 5마일 반경 이내의 구역을 정의하는 공간 데이터가 포함됩니다. 지오코더가 생성한 데이터는 LOCATION 컬럼에 놓입니다. 구역을 정의하는 데이터는 SALES_AREA 컬럼에 놓입니다.
   - 팀에서는 LOCATION 컬럼과 SALES_AREA 컬럼 두 개를 계층으로 등록합니다.
   - 팀에서 기본 지오코더를 사용하여 두 LOCATION 컬럼을 자동으로 관리합니다.
4. 공간 관리팀은 CUSTOMER 테이블의 LOCATION 컬럼, 전체 OFFICES 테이블 및 새로운 HAZARD_ZONES 테이블을 데이터 상주시킵니다.
   - 팀에서는 제어 센터를 사용하여 CUSTOMER 테이블의 LOCATION 컬럼을 데이터 상주시킵니다.
     a. 팀에서는 미국 Census Bureau의 레코드에 있는 주소와 그의 대응부가 100% 정확해야 한다는 조건하에서만 LOCATION 컬럼으로 주소용 공간 데이터를 삽입하도록 지오코더에 지시합니다(Census Bureau에서 제공하는 주소 파일은 DB2 Spatial Extender와 함께 제공됩니다. 지오코더가 소스 데이터의 주소를 공간 데이터로 변환하기 전에, 지오코더가 이

주소와 파일에 있는 그 대응부가 일치하는지 시도해야 합니다. 사용자는 공간 데이터를 테이블에 위치시키기 위해서 어느 정도 정확하게 일치해야 하는지 지정합니다. 이 백분율을 *정밀도*라고 합니다).

b. 팀에서는 지오코더를 일괄처리 모드로 실행하여 테이블의 모든 주소를 하나의 조작으로 지오코딩할 수 있습니다. 당황스럽게도 지오코더는 10개의 주소당 하나 정도를 거부합니다.

c. 팀에서는 거부한 레코드가 Census Bureaudml 레코드와 완전 일치하지 않는 새로운 주소라고 추측합니다. 그러한 문제를 해결하기 위해 팀에서는 정밀도를 85로 낮춥니다.

d. 팀에서는 지오코더를 일괄처리 모드로 다시 실행합니다. 주소를 거부하는 비율이 허용 가능한 레벨로 떨어졌습니다.

- 개별적인 GIS가 제공하는 유틸리티를 사용하여 팀은 사무소 데이터를 파일로 로드합니다. 그런 다음, 팀은 제어 센터를 사용하여 파일에서 새로운 OFFICES 테이블로 이 데이터를 가져오기합니다.
- 제어 센터를 사용하여 팀은 HAZARD ZONES 테이블을 작성하고 그의 공간 컬럼을 계층으로 등록하며 데이터를 그 컬럼으로 가져오기합니다. 데이터는 맵 벤더가 제공한 파일에서 옵니다.

5. 제어 센터를 사용하여 공간 관리팀은 새 계층에 대한 액세스를 촉진합니다.
   - 팀에서 그에 대한 색인을 작성합니다.
   - 팀에서 CUSTOMERS 및 HAZARD ZONES 테이블로부터 컬럼을 조인하는 뷰를 작성합니다. 그런 다음, 팀에서 뷰의 공간 컬럼을 계층으로 등록합니다.
6. 공간 분석팀은 조회를 실행하여 원래의 목적에 맞는 정보를 얻고, 새로운 지점 사무소를 설립할 위치를 결정하며 위험 지역에 대한 고객의 근접 기준에 따라 보험료를 조정합니다.

제2장 DB2 Spatial Extender 설치

이 장에서는 DB2 Spatial Extender를 설치하기 위한 지시사항을 제공합니다. 다음의 주제에 대해 설명합니다.

DB2 Spatial Extender 구성

DB2 Spatial Extender 시스템은 DB2 Universal Database, DB2 Spatial Extender 및 geobrowser(예: ArcExplorer)로 구성됩니다. 일반적으로 공간 데이터 조작에 사용되는 데이터베이스가 서버에 위치합니다. 사용자는 클라이언트 응용프로그램을 사용하여 DB2 Spatial Extender 저장 프로시듀어 및 공간 조회를 통해 공간 데이터에 액세스할 수 있습니다. 또한, geobrowser로 공간 데이터도 볼 수 있습니다.

20 페이지의 그림5에서는 DB2 Spatial Extender의 구조에 대해 설명합니다.

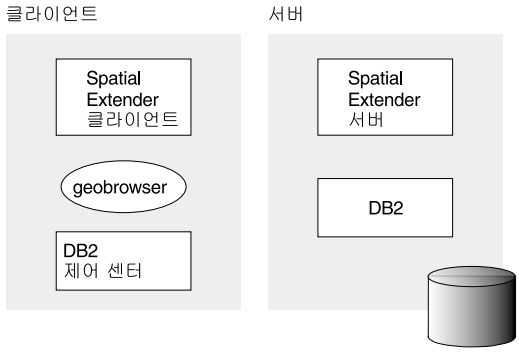


그림 5. 클라이언트/서버 설치

시스템 요구사항

이 절에서는 DB2 Spatial Extender에 대한 소프트웨어 및 하드웨어 요구사항에 대해 설명합니다.

지원되는 운영 체제

DB2 Spatial Extender는 다음과 같은 운영 체제에 설치될 수 있습니다.

- AIX 4.2 이상
- 서비스 팩 5가 있는 Windows NT 4.0 이상

필수 데이터베이스 소프트웨어

DB2 Spatial Extender를 설치하기 전에 시스템에 DB2 소프트웨어를 설치하여 구성해야 합니다. 21 페이지의 표1에서는 DB2 Spatial Extender 클라이언트 구성요소 및 DB2 Spatial Extender 서버 구성요소 양쪽에 대한 데이터베이스 소프트웨어 요구사항들을 나열합니다.

표 1. 최소 소프트웨어 요구사항

구성요소	소프트웨어
클라이언트	DB2 Administration Client, 버전 7.1[1]
서버	다음 중 하나: • DB2 Universal Database Enterprise Edition, 버전 7.1 • DB2 Universal Database Enterprise – Extended Edition, 버전 7.1[2]
	주: 1. 공간 데이터 또는 DB2 Spatial Extender 샘플 프로그램에 액세스하는 데 DB2 제어 센터 geobrowser를 사용할 계획이 *아니라면*, DB2 Administration Client의 다운레벨 버전을 사용할 수 있습니다. 2. DB2 Universal Database Enterprise - Extended Edition과 함께 DB2 Spatial Extender를 사용할지라도, 광범위 병렬 처리(MPP) 환경에서처럼 여러 노드에 걸쳐 공간 색인을 파티션할 수 없습니다.

디스크 공간 요구사항

표2에서는 DB2 Spatial Extender에 대한 바람직한 디스크 공간 요구사항을 나열합니다.

표 2. 디스크 공간 요구사항

DB2 Spatial Extender 구성요소	디스크 공간
DB2 Spatial Extender 서버 라이브러리(DB2 Spatial Extender 서버 라이브러리, 지오코더 참조 자료 및 문서 포함)	600 MB
DB2 Spatial Extender 클라이언트 지원(샘플 프로그램 데이터 포함)	15 MB

DB2 Spatial Extender 설치

이 절에서는 Windows NT 및 AIX 운영 체제에서 DB2 Spatial Extender를 설치하는 데 필요한 정보를 제공합니다.

시작하기 전에

아직 시작하지 않았으면, 클라이언트 워크스테이션에 DB2 Administration Client(제어 센터 및 런타임 클라이언트를 포함한 관리 도구)를 설치하고 DB2 Universal Database Enterprise Edition 또는 DB2 Universal Database Enterprise - Extended Edition을 설치하십시오. 그에 관련된 지시사항은 해당 *빠른 시작* 책에 있습니다.

Windows NT 시스템에 DB2 Spatial Extender 설치

Windows NT 시스템상에 DB2 Spatial Extender를 설치하려면 다음을 수행하십시오.

1. 필요한 관리 권한을 가진 사용자 이름으로 시스템에 로그온하십시오.
2. 다른 프로그램들을 시스템 종료(shut down)하십시오.
3. 드라이브에 CD-ROM을 삽입하십시오. 설치 런치패드가 열립니다.
4. 선택적: **릴리스 정보**를 클릭하여 설치 프로세스에 대한 변경사항에 관해 DB2 Spatial Extender 릴리스 정보를 점검한 후, DB2 Spatial Extender 발사대로 리턴하십시오.
5. **설치**를 클릭하십시오.
6. 설치 프로그램의 프롬프트에 응답하십시오. 나머지 단계에서는 온라인 도움말을 통해 안내를 받을 수 있습니다. 온라인 도움말을 호출하려면 **도움말**을 클릭한 후 F1 키를 누르십시오.

설치가 완료되면 DB2 Spatial Extender가 디렉토리 %DB2PATH%(예: c:\sqllib) 아래에 설치됩니다.

AIX 시스템에 DB2 Spatial Extender 설치

AIX 시스템상에 DB2 Spatial Extender를 설치하려면 다음을 수행하십시오.

1. 루트로 로그인하십시오.
2. 드라이브에 CD-ROM을 삽입하십시오.
3. AIX 시스템에서 CD-ROM을 마운트하십시오. *IBM UNIX용 DB2 Universal Database 빠른 시작*에서 CD-ROM 마운트에 대한 자세한 내용을 참조하십시오.
4. 다음 명령을 입력하여 CD-ROM이 마운트된 디렉토리로 변경하십시오.

```
cd /cdrom
```

여기서, *cdrom*은 AIX에서의 CD-ROM 드라이브의 마운트 지점입니다.

5. **db2setup** 명령을 입력하여 DB2 설치 프로그램을 시작하십시오. DB2 Spatial Extender 설치 창이 열립니다.

   주: DB2 설치 프로그램은 시스템에서 정보를 스캔하기 때문에 천천히 시작합니다.

6. DB2 Spatial Extender 설치 창의 제품 목록에서 설치할 제품을 선택한 후 **확인**을 클릭하십시오.

   DB2 Spatial Extender 설치시 보다 자세한 정보나 도움을 받으려면 **도움말**을 클릭하십시오.

설치가 완료되면 DB2 Spatial Extender가 /usr/lpp/db2_07_01 디렉토리에 설치됩니다.

설치 검증

DB2 Spatial Extender를 설치한 후 DB2 Spatial Extender 샘플 프로그램을 사용하여 설치를 검증할 수 있습니다. 샘플 프로그램을 실행하기 전에, 샘플 데이터베이스를 작성하고 샘플 프로그램을 실행해야 합니다.

주: DB2 Spatial Extender makefile에 지정된 컴파일러를 사용해야 합니다.

Windows NT용 샘플 프로그램을 컴파일 및 실행하려면 다음을 수행하십시오.

1. 관리자 권한을 보유한 사용자 ID로 로그인하십시오.
2. 명령행 프롬프트에서 **db2sampl**을 입력하고 DB2 샘플 데이터베이스를 작성하십시오.
3. 명령 행 프롬프트에서 다음 명령을 입력하십시오.

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

주: 3 단계를 수행하고 설치를 계속 검증하려면, 기본 DB2 인스턴스를 보유해야 합니다 (DB2–DB2).

4. **make rungsedemo**를 입력하십시오.

5. **rungsedemo.exe**를 입력하십시오.
6. 프로그램이 실행 중일 때 표시되는 오류 및 완료 메시지를 점검하십시오.

AIX용 샘플 프로그램을 컴파일 및 실행하려면 다음을 수행하십시오.

1. 루트로 로그인하십시오.
2. DB2 인스턴스를 작성하거나 갱신하십시오.
3. 명령행 프롬프트에서 **db2sampl**을 입력하고 DB2 샘플 데이터베이스를 작성하십시오.
4. 명령 행 프롬프트에서 다음 명령을 입력하십시오.

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

주: 4 단계를 수행하고 설치를 계속 검증하려면, 작성하거나 갱신한 기본 DB2 인스턴스를 보유해야 합니다.

5. **make rungsedemo**를 입력하십시오.
6. **rungsedemo**를 입력하십시오.
7. 프로그램이 실행 중일 때 표시되는 오류 및 완료 메시지를 점검하십시오.

65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 샘플 프로그램에 대한 자세한 정보를 참조하십시오.

사후설치 고려사항

DB2 Spatial Extender를 정상적으로 설치한 후에는 다음 사항을 고려해야 합니다.

- ArcExplorer의 다운로드
- DB2 인스턴스 갱신 유틸리티의 실행

ArcExplorer의 다운로드

IBM에서 ArcExplorer Java 3.0을 샘플 프로그램으로 배포하거나 http://www.esri.com의 ESRI 웹 사이트에서 이를 구할 수 있습니다.

*Using ArcExplorer*에서 ArcExplorer의 설치 및 사용에 대한 자세한 내용을 참조하십시오. 이 책은 ESRI 웹 사이트에서도 사용할 수 있습니다.

ArcExplorer에는 http://java.sun.com의 Sun 웹 사이트에서 아무 수수료없이 사용 가능한 Java® 2 런타임 환경(표준 개정판 또는 Enterprise Edition), V1.2.2가 필요합니다.

주: DB2 Universal Database V7.1은 IBM JDK 1.1.8과 함께 제공됩니다. ArcExplorer용 JRE 1.2.2를 설치할 때는 이를 DB2와 다른 디렉토리에 놓으십시오. CLASSPATH 환경 변수를 적절하게 설정하는 것에도 유념하십시오.

DB2 인스턴스 갱신 유틸리티(db2iupdt) 실행

db2iupdt 유틸리티는 지정된 DB2 인스턴스를 갱신하여 다음을 수행합니다.

- 인스턴스가 새로운 시스템 구성을 확보할 수 있도록 합니다.
- 인스턴스가 특정 제품 옵션의 설치 또는 제거와 연관된 함수에 액세스할 수 있도록 합니다.

AIX에서 이 유틸리티는 /usr/lpp/db2_07_01에 위치합니다. 도움이 필요하면 명령행에 `db2iupdt -h`를 입력하여 도움말 메뉴를 여십시오. Windows NT 운영 체제에서는 db2iupdt가 \sqllib\bin 디렉토리에 위치합니다. 그 디렉토리로 변경하여 명령을 입력하십시오. *IBM DB2 Universal Database Command Reference*에서 이 명령에 대한 자세한 내용을 참조하십시오.

다음 장

DB2 Spatial Extender를 설치했으면 DB2 제어 센터를 사용하여 GIS 환경을 설정하고 공간 정보에 대한 작업을 시작할 수 있습니다.

제어 센터로부터 DB2 Spatial Extender를 호출하려면 다음을 수행하십시오.

1. 제어 센터 창에서 DB2 Spatial Extender를 수행시킬 서버 아래의 **데이터베이스** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오.
2. **데이터베이스** 폴더를 클릭하십시오. 창의 오른쪽면에 있는 목차 분할창에 데이터베이스가 표시됩니다.
3. 작업하려는 데이터베이스를 마우스 오른쪽 버튼으로 클릭한 후, 수행하려는 공간 데이터 조작을 팝업 메뉴에서 클릭하십시오.

다음 장에서 제어 센터에서의 DB2 Spatial Extender 사용에 대한 자세한 내용을 참조하십시오.

- 27 페이지의『제3장 자원 설정』
- 39 페이지의『제4장 공간 컬럼 정의, 공간 컬럼을 계층으로 등록 및 지오코더를 사용하여 공간 컬럼을 유지보수』
- 47 페이지의『제5장 공간 컬럼 데이터 상주』
- 59 페이지의『제6장 공간 색인 작성』

제3장 자원 설정

DB2 Spatial Extender를 설치했으면, 공간 컬럼을 작성하고 공간 데이터를 관리할 때 필요한 자원이 있는 데이터베이스를 제공할 준비가 되었습니다. 이 장에서는 이러한 자원에 대해 요약하며 그 자원들을 사용 가능하게 만드는 두 가지 타스크 즉, 공간 데이터 조작용으로 데이터베이스를 사용하는 타스크와 공간 참조 시스템을 작성하는 타스크에 대해 설명합니다.

자원 재고

공간 컬럼을 작성하고 공간 데이터를 관리하기 위해 사용자가 발행하는 자원에는 다음이 포함됩니다.

- *참조 자료*: 지오코딩할 주소를 검증하기 위해 DB2 Spatial Extender가 점검하는 주소.
- 공간 데이터 조작용으로 데이터베이스를 사용하는 자원: 저장 프로시듀어, 공간 함수 및 기타.
- 사용자 및 벤더가 제공하는 기본 지오코더 이외의 지오코더들
- 공간 참조 시스템

이 절에서는 공간 데이터 조작용으로 데이터베이스를 사용하는 참조 자료 및 자원에 대해 설명합니다. 47 페이지의 『지오코딩 관련 정보』에서 기본값 이외의 geocoder에 대한 자세한 내용을 참조하십시오. 29 페이지의 『좌표 및 공간 참조 시스템 관련 정보』에서 공간 참조 시스템에 대한 자세한 내용을 참조하십시오.

참조 자료

*참조 자료*는 United States Census Bureau가 수집한 가장 최근의 주소로 구성됩니다. 기본 지오코더가 데이터베이스에 있는 주소를 좌표로 변환하기 전에 일부 또는 모든 주소를 참조 자료에 있는 주소로 먼저 일치시켜야 합니다.

참조 자료는 사용자가 DB2 Spatial Extender를 설치할 때 사용 가능하게 됩니다. 21 페이지의 『디스크 공간 요구사항』에서 이 데이터에 필요한 디스크 공간양에 대

한 내용을 참조하십시오. AIX에서 데이터가 제대로 로드되었는지 검증하려면 $DB2INSTANCE/sqllib/gse/refdata/ 디렉토리를 찾아 보십시오. Windows NT에서 데이터가 제대로 로드되었는지 검증하려면 %DB2PATH%\gse\refdata\ 디렉토리를 찾아 보십시오.

공간 데이터 조작용으로 데이터베이스를 사용하는 자원

DB2 Spatial Extender를 설치한 후 사용자가 수행하는 첫번째 타스크는 공간 데이터 조작용으로 데이터베이스를 사용 가능하게 하는 것입니다. 이에는 DB2 Spatial Extender가 다음과 같은 자원이 있는 데이터베이스를 로드하도록 조치를 시작하는 작업이 포함됩니다.

- 저장 프로시듀어. 제어 센터로부터 조치를 요청할 때, DB2 Spatial Extender가 이 저장 프로시듀어 중 하나를 호출하여 조치를 수행합니다.
- 공간 데이터 유형. 사용자는 공간 데이터가 저장될 각 테이블 또는 뷰 컬럼에 공간 데이터 유형을 지정해야 합니다. 자세한 내용은 39 페이지의 『공간 데이터 유형 관련 정보』를 참조하십시오.
- DB2 Spatial Extender의 카탈로그 테이블 및 뷰. 특정 조작들은 DB2 Spatial Extender 카탈로그에 따라 다릅니다. 예를 들어, 공간 데이터 유형이 있는 컬럼은 데이터 상주되기 전에 카탈로그에 계층으로서 등록되어야 합니다. 12 페이지의 『GIS 프로젝트 개발 및 구현』에서 계층에 대한 자세한 내용을 참조하십시오.
- 공간 색인 유형. 계층에 대해 색인을 정의할 수 있습니다.
- 공간 함수. 이를 사용하여 여러 가지 면에서 공간 데이터에 대해 작업할 수 있습니다. 예를 들어, 지리학적 기능 사이의 관계를 판별하고 공간 데이터를 추가로 생성하기 위해 작업할 수 있습니다. 이 기능 중 하나가 기본 지오코더입니다. 이는 주소를 좌표로 변환한 후에 이 좌표를 공간 컬럼에 삽입합니다. 133 페이지의 『제13장 기하학 및 관련 공간 함수』 및 173 페이지의 『제14장 SQL 조회에 대한 공간 함수』에서 공간 함수에 대한 자세한 내용을 참조하십시오. 47 페이지의 『지오코딩 관련 정보』에서 기본 지오코더에 대한 자세한 내용을 참조하십시오.
- 위에 나열된 오브젝트들이 들어 있는 DB2GSE라는 스키마.

『공간 데이터 조작용으로 데이터베이스를 사용』에서 제어 센터를 사용하여 이 자원들의 로드 작업을 시작하는 방법에 대한 내용을 참조하십시오. 65 페이지의 『제 8장 DB2 Spatial Extender용 응용프로그램 작성』에서 동일한 타스크를 수행하기 위한 응용프로그램 내의 루틴 사용에 대한 내용을 참조하십시오.

공간 데이터 조작용으로 데이터베이스를 사용

공간 데이터 조작용으로 데이터베이스를 사용하는 데 필요한 권한을 알아내려면 85 페이지의 『권한』을 참조하십시오.

제어 센터로부터 공간 데이터 조작용으로 데이터베이스를 사용 가능하게 하려면 다음을 수행하십시오.

1. 제어 센터 창에서 DB2 Spatial Extender를 수행시킬 서버 아래의 **데이터베이스** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오.
2. **데이터베이스** 폴더를 클릭하십시오. 창의 오른쪽면에 있는 목차 분할창에 데이터베이스가 표시됩니다.
3. 원하는 데이터베이스를 마우스 오른쪽 버튼으로 클릭한 후, 팝업 메뉴에서 **Spatial Extender —> 사용**을 클릭하십시오. DB2 Spatial Extender는 사용자가 공간 컬럼 및 데이터를 작성하고 그에 대해 작업할 수 있게 하는 자원들이 있는 데이터베이스를 공급합니다.

주: 공간 데이터 조작용으로 데이터베이스를 사용 가능하게 하기 전에 데이터베이스가 상주하는 서버에 DB2 Spatial Extender를 설치해야 합니다.

공간 참조 시스템 작성

이 절에서는 공간 참조 시스템과 좌표 시스템 사이의 관계 및 제어 센터로부터 공간 참조 시스템을 작성하는 방법에 대해 설명합니다.

좌표 및 공간 참조 시스템 관련 정보

이 절에서는 6 페이지의 『공간 데이터 특성』에서 시작된 좌표 시스템에 대해 계속해서 토론합니다. 그런 다음, 12 페이지의 『GIS 프로젝트 개발 및 구현』에 제공된 공간 참조 시스템의 정의로 진행됩니다. 또한 공간 참조 시스템의 매개변수에 지정할 값을 판별하기 위한 지침도 제공합니다.

좌표 시스템, 좌표 및 치수

지리적인 특정 영역을 다루는 가상 격자로 표현된 좌표 시스템을 고려해 볼 수 있습니다. 지구를 다루는 격자, 국가를 다루는 격자 또는 도의 영역을 다루는 격자가 이에 포함됩니다. 영역의 각 지리적인 기능은 동서 격자선과 남북 격자선의 교차점에 놓입니다. *X 좌표*라는 값은 동서 격자선상에 놓인 위치를 나타냅니다. *Y 좌표*라는 또다른 값은 남북 격자선상에 놓인 위치를 나타냅니다. 두 값 모두 격자의 중심 또는 *원점*에 대한 위치를 참조합니다.

원점에서의 X 및 Y 좌표는 둘 다 0입니다. 원점에서 동쪽으로 향하는 X 좌표는 양수이고 원점에서 서쪽으로 향하는 좌표는 음수입니다. 마찬가지로 원점에서 북쪽으로 향하는 Y 좌표는 양수이고 원점에서 서쪽으로 향하는 좌표는 음수입니다. 이러한 분산을 설명하기 위해서는 좌표 시스템 A에 광범위한 대도시 영역을 다루는 격자가 포함되어 있는 그러한 일반화된 예를 고려해 봅시다. X 좌표 7은 이 격자의 원점에서 동쪽으로 7개의 치수에 있는 위치를 나타냅니다. X 좌표 -9.5는 원점에서 서쪽으로 9개 반의 치수에 있는 위치를 나타냅니다.

공간 컬럼의 각 데이터 항목에는 지리적 기능의 위치를 정의하는 X 좌표 및 Y 좌표가 포함되거나, 기능의 일부 위치를 정의하거나 기능이 다루는 영역을 정의하는 다중 X 및 Y 좌표가 포함됩니다. 두 종류의 다른 값 즉, *Z 좌표*와 *치수*가 포함될 수도 있습니다. X 및 Y 좌표와 달리 Z 좌표 및 치수는 위치나 영역을 정의하기 위해 DB2 Spatial Extender에서 사용되지 않습니다. 오히려 이들은 GIS 응용프로그램에 필요한 정보를 단순히 전달합니다. Z 좌표는 일반적으로 지리적 기능의 높이나 깊이를 나타냅니다. 원점 위에 있는 Z 좌표는 양수이고 원점 아래에 있는 Z 좌표는 음수입니다. 치수는 숫자이며 어떤 종류의 정보도 전달할 수 있습니다. 예를 들어, GIS에서 유정을 나타낸다고 가정합시다. 응용프로그램에서 지진 데이터에 대한 진앙지 ID를 나타내는 값을 처리해야 할 경우에는, 이 값을 특정 단위로서 저장할 수 있습니다.

공간 참조 시스템, 옵셋 및 스케일 인수

『좌표 시스템, 좌표 및 치수』에 명시된대로 좌표값은 음수일 수 있으며 10진수로 표시됩니다. 치수의 경우에도 마찬가지입니다. 그러나 저장영역 오버헤드를 줄이기 위해 DB2 Spatial Extender는 각 좌표와 치수를 음수가 아닌 정수(즉 양수 또는 0)로 저장합니다. 그러므로, 실제 음수 및 10진 좌표와 치수들은 DB2 Spatial Extender가 이들을 저장할 수 있도록 음수가 아닌 정수로 변환되어야 합니다. 더

구나, 사용자는 DB2 Spatial Extender에 변환 방법을 알려 주어야 합니다. 이는 특정 매개변수를 설정하여 수행할 수 있습니다. 지리적인 특정 영역 내의 좌표와 치수로 변환하는 데 사용될 매개변수 설정값을 모두 집합해서 *공간 참조 시스템*이라고 합니다.

다음을 수행하여 공간 참조 시스템을 작성할 수 있습니다.

- 표현할 기능에 대한 가장 낮은 음의 좌표와 치수를 결정하여. (음수값은 0으로부터 멀수록 더 낮은 값입니다. X 좌표 –10이 X 좌표 –5보다 더 낮으며, 치수 –100이 치수 –50보다 더 낮습니다.)
- *옵셋 인수*(또는 간단히 말해 *옵셋*)를 지정하여. 음의 좌표에서 뺄 때 음이 아닌 숫자가 남는 값.
- *스케일 인수*를 지정하여. 10진 좌표 및 치수를 곱했을 때 정밀도가 적어도 좌표 또는 치수의 정밀도와 동일한 정수를 산출하는 값. 예를 들어, 정밀도 4인 좌표 92.77을 고려해 봅시다. 이 좌표에 스케일 인수 100을 곱해 정밀도 4인 정수 9277을 얻을 수 있습니다.

최저 음좌표와 치수를 결정

공간 참조 시스템에 대해 매개변수를 설정하기 전에, 정보를 원하는 기능이 들어 있는 지리적 영역 내의 최저 음수 X 좌표, Y 좌표, Z 좌표 및 치수를 판별해야 합니다. 다음 질문에 응답함으로써 이 값들을 알아낼 수 있습니다.

- 표현할 지형 중에서, 사용하려는 좌표 시스템의 원점 왼쪽에 놓여 있는 지형이 있습니까? 그러한 경우에 어떤 X 좌표가 가장 서쪽 지형의 위치 또는 서쪽 가장자리를 나타냅니까? (응답은 사용자가 다룰 음 X 좌표의 최저값입니다.) 예를 들어, 유정을 표현하는데 그 중 일부가 원점의 서쪽에 놓여 있는 경우에 어떤 X 좌표가 가장 왼쪽의 유정 위치를 나타냅니까?
- 원점의 남쪽에 지형이 있습니까? 그러한 경우에 어떤 Y 좌표가 가장 남쪽 지형의 위치 또는 남쪽 가장자리를 나타냅니까? (응답은 사용자가 다룰 음 Y 좌표의 최저값입니다.) 예를 들어, 유정을 표현하는데 그 중 일부가 원점의 남쪽에 놓여 있는 경우에 어떤 Y 좌표가 가장 남쪽의 유정 위치를 나타냅니까?
- 깊이를 정의하는 데 Z 좌표를 사용할 경우에 어느 지형이 가장 깊고 어떤 Z 좌표가 지형의 가장 낮은 지점을 나타냅니까? (응답은 사용자가 다룰 음 Z 좌표의 최저값입니다.)

- 공간 데이터에 치수를 포함시킬 경우에 음수값이 있습니까? 그러한 경우에 음 치수의 최저값은 무엇입니까?

최저 음 좌표 및 치수를 확인했으면, 각각에 그 값의 5-10%에 해당하는 양을 추가하십시오. 예를 들어, 최저 음 X 좌표가 –100이면 그 값에 –5를 추가할 수 있습니다. 이 책에서는 결과 그림을 *인수화값*(augmented value)이라고 합니다.

옵셋 인수 지정

그 다음에 DB2 Spatial Extender가 음 좌표 및 치수를 음수가 아닌 값으로 변환하는 데 사용해야 하는 옵셋 인수를 지정하십시오.

- 증가되는 X 값이 어떤 값이 되기를 원하는지 결정한 뒤, 이 값에서 뺄 때 0이 남는 옵셋을 지정하십시오. 그런 다음, DB2 Spatial Extender는 양수값에 도달하기 위해 모든 음의 X 좌표로부터 이 숫자를 뺍니다. DB2 Spatial Extender는 또한 다른 모든 X 좌표에서 이 숫자를 뺍니다.

  예를 들어, 인수화된 X 값이 –105이면 0을 얻기 위해 –105를 빼야 합니다. 그런 다음, DB2 Spatial Extender는 사용자가 표현할 기능과 연관된 모든 X 좌표에서 –105를 뺍니다. 이 좌표 중 어느 것도 –100보다 크지 않기 때문에, 모든 빼기 결과 값이 양수가 됩니다.
- 마찬가지로 인수화된 Y 값, 인수화된 Z 값 및 인수화된 치수에서 뺄 때 0이 남는 옵셋을 지정하십시오.

X 좌표에서 뺀 옵셋을 *거짓 X*라고 합니다. Y 좌표, Z 좌표 및 치수에서 뺀 옵셋을 각각 *거짓 Y*, *거짓 Z* 및 *거짓 M*이라고 합니다. 34 페이지의 『제어 센터로부터 공간 참조 시스템 작성』에서 제어 센터로부터 이 매개변수를 지정하는 지시사항을 참조하십시오.

스케일 인수 지정

그 다음에 DB2 Spatial Extender가 10진 좌표 및 치수를 정수로 변환하는 데 사용해야 하는 스케일 인수를 지정하십시오.

- 10진 X 좌표 또는 10진 Y 좌표로 곱할 때 32비트 정수를 산출하는 스케일 인수를 지정하십시오. 이 스케일 인수를 10의 인수로 만드는 것이 바람직합니다. 즉, 10의 1승(10), 10의 2승(100), 10의 3승(1000) 또는 필요하다면 보다 큰 인수를 사용할 수 있습니다. 스케일 인수가 10의 몇승이 되어야 하는지 판별하려면 다음을 수행하십시오.

1. 어느 X 및 Y 좌표가 십진수인지 또는 십진수가 될 것 같은지 판별하십시오. 예를 들어, 사용자가 다룰 다양한 X 및 Y 좌표 중에서 세 개가 십진수(1.23, 5.1235 및 6.789)임을 판별했다고 가정하십시오.
2. 가장 긴 10진 정밀도를 가진 10진 좌표를 취하십시오. 그런 다음, 동일한 정밀도의 정수를 산출하기 위해 이 좌표에 곱할 수 있는 10의 인수가 무엇인지를 판별하십시오. 즉, 이 예의 경우에는 세 개의 10진 좌표 중에서 5.1235가 가장 긴 십진 정밀도를 가집니다. 이 좌표에 10의 4승(10000)을 곱하면 정수 51235가 생성됩니다.
3. 방금 설명한 곱셈에 의해 생성된 정수가 너무 길어 32비트 데이터 항목으로 저장할 수 없는지 여부를 판별하십시오. 51235는 너무 길다고 볼 수 없습니다. 그러나, 1.23, 5.11235 및 6.789 이외에 X 및 Y 좌표의 범위에 네 번째 값 10006.789876이 포함된다고 가정하십시오. 이 좌표의 십진 정밀도가 다른 세 개의 정밀도 보다 길기 때문에, *이* 좌표(5.1235가 아님)에 10의 인수를 곱합니다. 이를 정수로 변환하기 위해 이 좌표에 10의 6승(1000000)을 곱할 수 있습니다. 그러나 결과값 10006789876이 너무 길어 32비트 데이터 항목으로 저장할 수 없습니다. DB2 Spatial Extender가 이를 저장하려고 시도하면 예기치 못한 결과가 발생합니다.

   이러한 문제를 없애려면, 원래 좌표로 곱할 때, DB2 Spatial Extender가 최소한의 정밀도 손실을 보면서 저장 가능한 정수로 절단할 수 있는 그러한 십진수를 생성하는 10의 인수를 선택하십시오. 이 경우에는 10의 4승(10000)을 선택할 수 있습니다. 10000에 10006.789876을 곱하면 100067898.76이 됩니다. DB2 Spatial Extender는 실제적으로 매우 사소한 양의 정확도를 감소시키면서 이 숫자를 100067898로 절단합니다.

- 표현할 지형에 십진 Z 좌표가 있으면, 앞에서 설명한 프로시듀어를 따라 이 좌표에 대한 *스케일 인수*를 확인하십시오. 지형에 십진 치수가 연관되어 있으면, 이 동일한 프로시듀어를 따라 이 치수에 대한 *스케일 인수*를 확인하십시오.

X 및 Y 좌표에 대한 *스케일 인수*를 *XY 단위*라고 합니다. Z 좌표 및 치수에 대한 *스케일 인수*를 각각 *Z 단위* 및 *M 단위*라고 합니다. 34 페이지의 『제어 센터로부터 공간 참조 시스템 작성』에서 제어 센터로부터 이 매개변수를 지정하는 지시사항을 참조하십시오.

제어 센터로부터 공간 참조 시스템 작성

이 절에서는 제어 센터로부터 공간 참조 시스템을 작성하는 단계에 대한 개요를 제공합니다. 이 개념 뒤에 각 단계를 완성하는 방법에 대한 세부사항들이 나옵니다.

이 단계를 수행하는 데 아무 권한도 필요하지 않습니다.

제어 센터로부터의 공간 참조 시스템 작성 단계에 대한 개요:

1. 공간 참조 작성 창을 여십시오.
2. 사용하려는 좌표 시스템을 지정하십시오.
3. 작성하려는 공간 참조 시스템에 대한 식별자를 지정하십시오.
4. 관련 정보를 원하는 지리적 기능에 적용되는 좌표 범위와 치수를 판별하십시오.
5. 음수 또는 10진수 좌표 및 치수를 DB2 Spatial Extender가 저장할 수 있는 데이터 항목으로 변환하는 데 사용될 수 있는 값을 지정하십시오.
6. DB2 Spatial Extender에게 지시하여 원하는 공간 참조 시스템을 작성하십시오.

제어 센터로부터의 공간 참조 시스템 작성 단계에 대한 세부사항:

1. 공간 참조 작성 창을 여십시오.
   a. 제어 센터 창에서 DB2 Spatial Extender를 수행시킬 서버 아래의 **데이터베이스** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오.
   b. **데이터베이스** 폴더를 클릭하십시오. 창의 오른쪽면에 있는 목차 분할창에 데이터베이스가 표시됩니다.
   c. 공간 데이터에 사용하는 데이터베이스를 마우스 오른쪽 버튼으로 클릭한 후, 팝업 메뉴에서 **Spatial Extender —> 공간 참조**를 클릭하십시오. 공간 참조 창이 열립니다.
   d. 공간 참조 창에서 **작성**을 클릭하십시오. 공간 참조 작성 창이 열립니다.
2. 공간 참조 작성 창에서 **좌표 시스템** 필드를 사용하여 어느 좌표 시스템을 사용할 것인지 지정하십시오.
3. 작성하려는 공간 참조 시스템에 대한 식별자를 지정하십시오.

- **이름** 필드에 시스템 이름으로 1-64자를 입력하십시오.

  제한사항: 다른 공간 참조 시스템의 이름을 지정하지 마십시오. 데이터베이스에 있는 두 개의 공간 참조 시스템 어느 것도 동일한 이름을 가질 수 없습니다.

- ID 필드에 숫자 식별자를 입력하십시오. 이는 정수이어야 합니다.

  제한사항: 다른 공간 참조 시스템의 ID를 지정하지 마십시오. 데이터베이스에 있는 두 개의 공간 참조 시스템 어느 것도 동일한 ID를 가질 수 없습니다.

4. 제어 센터 이외의 매체를 사용하여(예: 종이 또는 칠판) 사용자가 나타낼 지리적 기능에 적용되는 가장 낮은 음의 좌표와 치수를 판별하십시오. 31 페이지의 『최저 음좌표와 치수를 결정』에서 이를 수행하는 방법에 대한 자세한 내용을 참조하십시오.
5. 공간 참조 작성 창에서 음수 또는 10진수 좌표 및 치수를 DB2 Spatial Extender가 지원하는 데이터 항목 즉, 음수가 아닌 32비트 정수로 변환하기 위한 값을 지정하십시오.
   a. 음수 또는 10진수 좌표를 음수가 아닌 정수로 변환하기 위한 값을 지정하십시오.
      - **옵셋** 컬럼에서 **X**에 가장 가까운 필드에 거짓 X를 지정하십시오.
        - 단계 4에 나타난 X 좌표 범위 내의 값이 음수이면, 가장 낮은 음좌표에서 빼기할 때 양수가 남게 되는 거짓 X를 입력하십시오. 32 페이지의 『옵셋 인수 지정』에서 자세한 내용을 참조하십시오.
        - 모든 X 좌표가 음수가 아니면, 거짓 X인 0을 입력하십시오.
      - **스케일 인수** 컬럼에서 **X**의 가장 오른쪽에 있는 필드에 XY 단위를 지정하십시오. 이 XY 단위는 10진수 X 좌표 또는 10진수 Y 좌표로 곱하기 했을 때 최소한의 정밀도 손상을 감안해서 32비트 데이터 항목으로 저장될 수 있는 정수를 산출해야 합니다. 32 페이지의 『스케일 인수 지정』에서 자세한 내용을 참조하십시오.

        X의 가장 오른쪽에 있는 필드에 XY 단위를 지정하고 나면, 그 값이 **Y**의 가장 오른쪽에 있는 필드에도 나타납니다.

b. DB2 Spatial Extender가 음의 Y 좌표를 양수값으로 변환할 수 있도록 거짓 Y를 지정하십시오. **Y**와 가장 가까운 필드에 있는 **옵셋** 컬럼에서 이를 수행합니다.

   - 단계 35 페이지의 4에 나타난 Y 좌표 범위 내의 값이 음수이면, 가장 낮은 음좌표에서 빼기할 때 양수가 남게 되는 거짓 Y를 입력하십시오. 32 페이지의 『옵셋 인수 지정』에서 자세한 내용을 참조하십시오.
   - 모든 Y 좌표가 양수이면 거짓 Y인 0을 입력하십시오.

c. 공간 데이터에 Z 좌표를 포함시킬 경우에 음수 또는 10진수 Z 좌표를 음수가 아닌 정수로 변환하기 위한 값을 지정하십시오.

   - **옵셋** 컬럼에서 **Z**에 가장 가까운 필드에 거짓 Z을 입력하십시오.
     - 단계 35 페이지의 4에 나타난 Z 좌표 범위 내의 값이 음수이면, 가장 낮은 음좌표에서 빼기할 때 양수가 남게 되는 거짓 Z을 입력하십시오. 32 페이지의 『옵셋 인수 지정』에서 자세한 내용을 참조하십시오.
     - 모든 Z 좌표가 음수가 아니면, 거짓 Z인 0을 입력하십시오.
   - **스케일 인수** 컬럼에서 **Z**의 가장 오른쪽에 있는 필드에 Z 단위를 지정하십시오. 이 Z 단위는 10진수 Z 좌표로 곱하기 했을 때 최소한의 정밀도 손상을 감안해서 32비트 데이터 항목으로 저장될 수 있는 정수를 산출해야 합니다. 32 페이지의 『스케일 인수 지정』에서 자세한 내용을 참조하십시오.

d. 공간 데이터에 치수를 포함시킬 경우에 음수 또는 10진수 특정 단위를 양수로 변환하기 위한 값을 지정하십시오.

   - **옵셋** 컬럼에서 **선형** 레이블에 가장 가까운 필드에 거짓 M을 입력하십시오.
     - 단계 35 페이지의 4에 나타난 치수 범위 내의 값이 음수이면, 가장 낮은 음 치수에서 빼기할 때 양수가 남게 되는 거짓 M을 입력하십시오. 32 페이지의 『옵셋 인수 지정』에서 자세한 내용을 참조하십시오.
     - 모든 치수가 양수이면 거짓 M인 0을 입력하십시오.

- **스케일 인수** 컬럼에서 **선형** 레이블의 가장 오른쪽에 있는 필드에 M 단위를 지정하십시오. 이 M 단위는 10진수 치수로 곱하기 했을 때 최소한의 정밀도 손상을 감안해서 32비트 데이터 항목으로 저장될 수 있는 정수를 산출해야 합니다. 32 페이지의 『스케일 인수 지정』에서 자세한 내용을 참조하십시오.

6. **확인**을 클릭하여 원하는 공간 참조 시스템을 작성하십시오.

제4장 공간 컬럼 정의, 공간 컬럼을 계층으로 등록 및 지오코더를 사용하여 공간 컬럼을 유지보수

DB2 Spatial Extender GIS에 대한 자원을 설정한 후에는 공간 데이터를 포함할 오브젝트를 작성할 준비가 되었습니다. 예를 들어, 공간 데이터를 포함할 새 테이블이 필요한 경우에는, 데이터가 들어갈 컬럼에 공간 데이터 유형을 지정하여 이들을 정의할 수 있습니다. 기존 테이블에 공간 컬럼을 추가해야 할 경우에도 수행할 수 있습니다.

공간 컬럼이 있는 신규 또는 기존 테이블을 제공할 때, 이 컬럼을 계층으로 등록해야 합니다. 또한, 지오코더가 컬럼을 데이터 상주시키도록 할 계획이면 사용자가 컬럼을 계층으로 등록할 때 지오코더가 이를 자동으로 관리하게 할 수 있습니다. 이는 다음과 같은 방법으로 발생됩니다. 즉, DB2 Spatial Extender는 공간 컬럼의 해당 속성 컬럼(들)이 신규 또는 갱신된 데이터를 수신할 때마다 지오코더를 호출하기 위해 코드화된 트리거를 정의합니다. 호출될 때 지오코더는 신규 또는 갱신된 데이터를 공간 데이터로 변환하고 이 공간 데이터를 공간 컬럼에 위치시킵니다.

테이블에 대한 공간 컬럼을 정의한 후에, 사용자의 선택에 따라 이 테이블 컬럼을 통해 뷰 컬럼을 작성할 수 있습니다. 테이블 컬럼을 계층으로 등록한 후에 뷰 컬럼을 계층으로 등록해야 합니다.

이 장에서는 사용자가 공간 컬럼에 지정할 수 있는 데이터 유형의 특성 및 사용에 대해 설명합니다. 또한, 제어 센터를 사용하여 테이블에 대한 공간 컬럼을 정의하고 이 컬럼을 계층으로등록하며 지오코더를 사용하여 이 컬럼을 유지보수하는 방법에 대해 설명합니다. 마지막으로 이 장에서는 제어 센터를 사용하여 뷰 컬럼을 계층으로 등록하는 방법에 대해 설명합니다.

공간 데이터 유형 관련 정보

이 절에서는 공간 컬럼에 필요한 데이터 유형에 대해 소개하고 공간 컬럼의 데이터 유형을 선택하기 위한 지침을 제공합니다.

공간 데이터 조작에 데이터베이스를 사용할 때, DB2 Spatial Extender는 구조화된 데이터 유형의 계층 구조를 가진 데이터베이스를 제공합니다. 그림6에서 이 계층 구조를 보여 줍니다. 이 그림에서 인스턴스화 가능 유형은 흰색 바탕이며 인스턴스화 불가능 유형은 회색 바탕입니다.

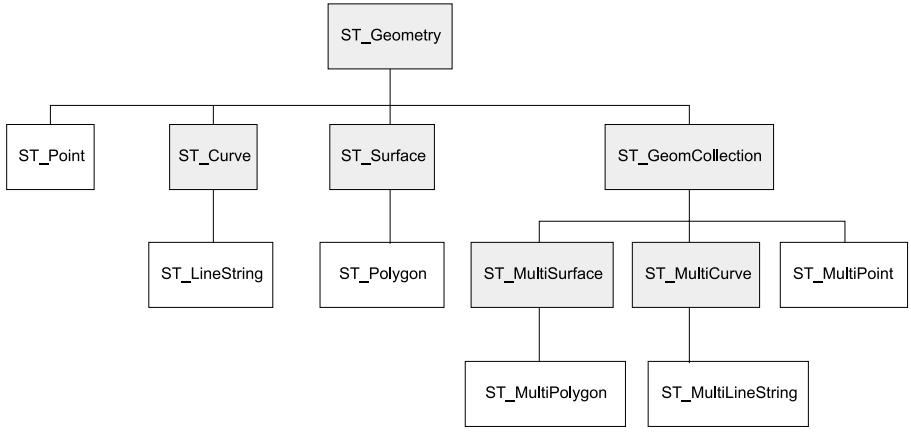


그림 6. 공간 데이터 유형의 계층 구조. 흰색 상자에 명명된 데이터 유형은 인스턴스화 가능합니다. 회색 상자에 명명된 데이터 유형은 인스턴스화 불가능합니다.

그림6의 계층 구조에는 다음이 포함됩니다.

- 단일 단위를 형성한 것으로 인식될 수 있는 지리 대상물에 대한 데이터 유형(예: 개별 주택지 및 고립된 호수).
- 다중 단위 또는 구성요소로 구성된 지리 대상물에 대한 데이터 유형(예: 도로 체계 및 산맥).
- 모든 종류의 지리 대상물에 대한 데이터 유형.

단일 단위 지형에 대한 데이터 유형

ST_Point, ST_Linestring 및 ST_Polygon을 사용하여, 단일 단위를 형성한 것으로 인식될 수 있는 지형이 점유한 공간을 정의하는 좌표를 저장하십시오.

- 이산 지리 대상물이 점유한 공간에 점을 지정할 때 ST_Point를 사용하십시오. 지형은 우물과 같이 매우 작을 수도 있고 도시와 같이 매우 클 수도 있고 빌딩 또는 공원과 같이 중간 크기의 지형일 수도 있습니다. 각각의 경우에 공간에 있는 점은 동서 좌표선(예: 위선) 및 남북 좌표선(예: 자오선)의 교차점에 놓여 있

을 수 있습니다. ST_Point 데이터 항목에는 그와 같은 교차점을 정의하는 값(X 좌표 및 Y 좌표)이 포함됩니다. X 좌표는 교차점이 동서쪽 선에 놓인 위치를 말하며, Y 좌표는 교차점이 남북선에 놓인 위치를 말합니다.

- 선형 지형이 점유한 공간을 정의하는 좌표에 대해 ST_Linestring을 사용하십시오(예: 거리, 운하 및 송수관).
- 다중 측면 지형으로 덮힌 공간 extent를 지정할 때 ST_Polygon을 사용하십시오(예: 복지 구역, 숲, 야생동물 서식지). ST_Polygon 데이터 항목은 그러한 지형의 주계를 정의하는 좌표로 구성됩니다.

  어떤 경우에는 ST_Polygon과 ST_Point가 동일한 지형에 사용될 수 있습니다. 예를 들어, 여러 아파트 단지에 관한 공간 정보가 필요하다고 가정합시다. 각 단지가 놓여 있는 공간에 점을 표현할 경우에는, ST_Point를 사용하여 그와 같은 각 점을 정의하는 X 및 Y 좌표를 저장합니다. 한편, 각 단지에 속해 있는 영역을 표현할 경우에는, ST_Polygon을 사용하여 그와 같은 각 영역의 주계를 정의하는 좌표를 저장합니다.

다중 단위 지형에 대한 데이터 유형

ST_MultiPoint, ST_MultiLineString 및 ST_MultiPolygon을 사용하여 다중 단위로 구성된 지형이 점유한 공간을 정의하는 좌표를 저장하십시오.

- 이산 단위로 구성된 지형을 표현할 때 그리고 각 구성요소가 점유한 공간에 점을 지정할 때 ST_MultiPoint를 사용하십시오. ST_MultiPoint 데이터 항목에는 그러한 지형의 각 요소에 대한 위치를 정의하는 X 및 Y 좌표의 쌍이 포함됩니다. 예를 들어, 행은 섬 군도를 나타내고 컬럼에는 ST_MultiPoint 컬럼이 포함된 테이블을 고려해 봅시다. 이 컬럼에 있는 각 데이터 항목에는 각 군도의 섬 위치를 정의하는 X 및 Y 좌표 쌍이 포함됩니다.
- 선형 단위로 구성된 지형을 표현할 때 그리고 각 단위가 점유한 공간에 관한 정보를 원할 때 ST_MultiLineString을 사용하십시오. ST_MultiLineString 데이터 항목은 그러한 공간을 정의하는 좌표로 구성됩니다. 예를 들어, 행은 강 체계를 나타내고 컬럼에는 ST_MultiLineString 컬럼이 포함된 테이블을 고려해 봅시다. 이 컬럼에 있는 각 데이터 항목에는 각 체계의 강 경로를 정의하는 좌표 세트가 포함됩니다.

- 다중 평면 단위로 구성된 지형을 표현할 때 그리고 각 단위가 점유한 공간에 관한 정보를 원할 때 ST_MultiPolygon을 사용하십시오. 예를 들어, 행은 중서부 지역을 나타내고 컬럼에는 ST_MultiPolygon 컬럼이 포함된 테이블을 고려해 봅시다. 이 컬럼에는 농장에 관한 정보가 들어 있습니다. 특히, 컬럼에 있는 각 데이터 항목에는 특정 도의 농장 주계를 정의하는 좌표 세트가 포함됩니다.

모든 지형에 대한 데이터 유형

기타 다른 데이터 유형 중 어느 것을 사용할 것인지 확신하지 못할 때 ST_Geometry를 사용할 수 있습니다. 이는 ST_Geometry가 다른 데이터 유형들이 속해 있는 계층 구조의 루트이기 때문에, ST_Geometry 컬럼은 다른 데이터 유형이 지정된 컬럼에 저장될 수 있는 값의 일부 또는 모두를 저장할 수 있습니다.

주의: 기본 지오코더는 주소를 ST_Point 또는 ST_Geometry 데이터 항목으로 변환할 수 있습니다. 그러므로, 이 지오코더를 사용하여 공간 컬럼을 데이터 상주시킬 계획이면, ST_Point 또는 ST_Geometry 데이터 유형을 이 컬럼에 지정해야 합니다.

테이블에 대한 공간 컬럼 정의, 이 컬럼을 계층으로 등록 및 지오코더를 사용하여 이 컬럼을 유지보수

이 절에서는 테이블에 대해 공간 컬럼을 정의하고 이 컬럼을 계층으로 등록하며 지오코더를 사용하여 이 컬럼을 유지보수하기 위한 단계들의 개요를 설명합니다. 이 개념 뒤에 각 단계를 완성하는 방법에 대한 세부사항들이 나옵니다.

테이블 컬럼을 계층으로서 등록하는 데 필요한 권한을 알아내려면 99 페이지의 『권한』을 참조하십시오. 지오코더를 사용하여 이 컬럼을 유지보수하는 데 필요한 권한을 알아내려면 82 페이지의 『권한』을 참조하십시오.

테이블에 대해 공간 컬럼을 정의하고 이 컬럼을 계층으로 등록하며 지오코더를 사용하여 이 컬럼을 유지보수하기 위한 단계의 개요:

1. 공간 컬럼이 새로운 테이블의 일부분인 경우에는 이 테이블을 작성하십시오.
2. 공간 계층 작성 창을 여십시오.

3. 테이블에 공간 컬럼을 추가한 후 이 컬럼을 계층으로서 등록할 것임을 지정하거나 기존 컬럼을 계층으로 등록할 것임을 지정하십시오.
4. 계층에 사용될 공간 참조 시스템을 지정하십시오.
5. 가져오기된 데이터 또는 다른 공간 컬럼으로부터 생성된 데이터가 계층에 포함되어 있는 경우에, 계층을 작성하려면 DB2 Spatial Extender에게 확인하십시오.
6. 속성 데이터로부터 파생된 데이터가 계층에 포함된 경우,
   a. 이 속성 데이터가 포함된 컬럼(들)을 지정하십시오.
   b. 지오코더를 사용하여 계층을 유지보수할 것임을 지정하십시오.
   c. 계층을 작성하려면 DB2 Spatial Extender에게 확인하십시오.

테이블에 대해 공간 컬럼을 정의하고, 이 컬럼을 계층으로 등록하며, 이를 유지보수하기 위해 지오코더를 사용하기 위한 세부 단계:

1. 공간 컬럼이 새로운 테이블의 일부분인 경우에는 이 테이블을 작성하십시오.
   - 선택항목의 인터페이스(예: 제어 센터 또는 명령행 프로세서)를 사용하여 테이블을 작성하십시오.
   - 지오코더를 사용할 계획이면 지오코더가 조작할 컬럼을 1 - 10개 정도 포함시키십시오. 지오코더는 입력으로 10개 이상의 데이터 컬럼을 취할 수 없습니다.
   - 계층으로 등록할 공간 컬럼을 포함시키거나 단계 44 페이지의 3에서 이 컬럼을 정의하십시오.

   기존 테이블을 사용하려면 다음 단계로 가십시오.
2. 공간 계층 작성 창을 여십시오.
   a. 제어 센터 창에서 공간 데이터 조작에 사용하는 데이터베이스 내의 테이블에 대해 **테이블** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오.
   b. **테이블** 폴더를 클릭하십시오. 창의 오른쪽면에 있는 목차 분할창에 테이블이 표시됩니다.
   c. 원하는 테이블을 마우스 오른쪽 버튼으로 클릭한 후 팝업 메뉴에서 **Spatial Extender —> 공간 계층**을 클릭하십시오. 공간 계층 창이 열립니다.
   d. 공간 계층 창에서 **작성**을 클릭하십시오. 공간 계층 작성 창이 열립니다.

3. 공간 계층 작성 창에서 테이블에 공간 컬럼을 추가한 후 이 컬럼을 계층으로서 등록할 것인지 지정하거나 기존 컬럼을 계층으로 등록할 것인지 지정하십시오.
   - 테이블에 공간 컬럼을 추가한 후 이 컬럼을 계층으로서 정의하려면 다음을 수행하십시오.
     a. **계층 컬럼** 필드에서 컬럼에 대한 이름을 입력하십시오.
     b. **컬럼 유형** 필드에, 컬럼이 가질 데이터 유형을 선택하거나 입력하십시오. 39 페이지의 『공간 데이터 유형 관련 정보』에서 허용 가능한 데이터 유형에 대한 내용을 참조하십시오.
   - 기존 컬럼을 계층으로서 정의하려면 **계층 컬럼** 필드에 이를 선택하십시오.

   제한사항: 이미 계층으로 정의된 컬럼은 선택하지 마십시오.
4. **공간 참조 시스템** 필드에 계층에 사용될 공간 참조 시스템의 이름을 지정하십시오.
5. 가져오기된 데이터 또는 다른 공간 컬럼으로부터 생성된 데이터가 계층에 포함되게 하려면, **확인**을 클릭하여 이를 등록하십시오.
6. 속성 데이터로부터 파생된 데이터가 계층에 포함되게 하려면 다음을 수행하십시오.
   a. 이 속성 데이터가 포함된 컬럼(들)을 지정하십시오.
      1) **사용 가능한 컬럼** 상자에서 컬럼(들)을 선택하십시오. 최대 10개의 컬럼을 선택할 수 있습니다.
      2) **>** 누름 버튼, **>>** 누름 버튼 또는 둘 다를 클릭하여 **선택한 컬럼** 상자에 있는 선택한 컬럼(들)을 나열하십시오.
   b. 지오코더를 사용하여 계층을 유지보수하려는 경우,
      1) **자동 지오코더 사용** 선택란을 선택하십시오.
      2) **이름** 필드에 사용하려는 지오코더의 이름을 선택하십시오.
      3) **정밀도 레벨** 필드에, 입력 레코드가 처리될 참조 자료에 있는 해당 레코드와 어느 정도 일치해야 하는 지를 지정하십시오(백분율 단위). 이 백분율을 *정밀도*라고 합니다. 예를 들어, 지오코더가 557 Bailey, San Jose 94120 주소가 포함된 입력 레코드를 읽는다고 가정합시다. 정밀

도가 100이면서 이 주소와 참조 자료의 상대방 사이에 일치하는 정도가 100% 정확하지 않은 경우에는, 지오코더가 이를 거부합니다. 정밀도가 75이면서 레코드와 그의 참조 자료 상대방 사이에 일치하는 정도가 최소 75%의 정확성을 가질 경우에는, 지오코더가 이를 처리합니다.

4) 벤더에서 제공한 지오코더인 경우에는 **등록 정보** 상자를 사용하여 사용하려는 벤더 제공 지오코딩 매개변수를 지정하십시오.

c. **확인**을 클릭하여 선택한 컬럼을 계층으로 등록하고, 사용자의 요청이 있다면 지오코더를 사용하여 컬럼을 유지보수하십시오.

뷰 컬럼을 계층으로 등록

뷰 컬럼을 계층으로서 등록하는 데 필요한 권한을 알아내려면 99 페이지의『권한』을 참조하십시오.

뷰 컬럼을 계층으로 등록하려면 다음을 수행하십시오.

1. 공간 계층 작성 창을 여십시오.
   a. 제어 센터 창에서 공간 데이터 조작에 사용하는 데이터베이스 내의 뷰에 대해 **뷰** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오.
   b. **뷰** 폴더를 클릭하십시오. 창의 오른쪽면에 있는 목차 분할창에 뷰가 표시됩니다.
   c. 원하는 뷰를 마우스 오른쪽 버튼으로 클릭한 후 팝업 메뉴에서 **Spatial Extender —> 공간 계층**을 클릭하십시오. 공간 계층 창이 열립니다.
   d. 공간 계층 창에서 **작성**을 클릭하십시오. 공간 계층 작성 창이 열립니다.
2. **계층 컬럼** 상자를 사용하여 계층으로 등록하려는 컬럼을 지정하십시오.
3. **기초 공간 계층** 필드에, 선택한 뷰 컬럼이 기초가 되는 테이블 컬럼의 이름을 지정하십시오. 이 테이블 컬럼은 계층으로 이미 등록된 상태이어야 합니다.
4. **확인**을 클릭하여 지정된 뷰 컬럼을 계층으로 등록하십시오.

제5장 공간 컬럼 데이터 상주

공간 컬럼을 계층으로 등록했으면 공간 데이터가 있는 공간 컬럼을 공급할 준비가 된 것입니다. 7 페이지의 『공간 데이터의 발원지』에 명시된 대로 이 데이터를 공급하는 데는 세 가지 방법 즉, 지오코더라는 기능을 사용하여 속성 데이터에서 공간 데이터를 유도하는 방법, 다른 기능들을 사용하여 다른 공간 데이터에서 이 공간 데이터를 유도하는 방법 또는 파일에서 공간 데이터를 가져오기하는 방법이 있습니다. 이 장에서는 다음 사항에 대해 다룹니다.

- 지오코딩에 대해 설명하고 제어 센터를 사용하여 속성 데이터를 일괄처리 모드로 지오코딩하는 방법에 대해 설명합니다.
- 이 절에서는 데이터의 가져오기 및 내보내기에 대해 설명하고 제어 센터를 사용하여 GIS로 데이터를 가져오기하고 GIS로부터 데이터를 내보내기하는 방법에 대해 설명합니다.

162 페이지의 『기존 기하학에서 새 기하학을 생성하는 함수』에서 기존 공간 데이터로부터 새로운 공간 데이터를 유도하는 기능에 대한 내용을 참조하십시오.

지오코더 사용

이 절에서는 지오코딩 프로세스에 대해 설명하고 제어 센터에서 일괄처리 모드로 지오코더를 수행하는 방법에 대해 설명합니다.

지오코딩 관련 정보

이 절에서는 지오코더와 그 소스 사이의 기본적인 차이점을 구별해 봅니다. 이는 지오코더가 작동할 수 있는 두 가지 모드에 대해서도 설명하고 지오코더의 사용 시기를 결정하는 데 고려해야 할 요인들도 소개합니다.

DB2 Spatial Extender로 다음을 수행할 수 있습니다.

- DB2 Spatial Extender와 함께 제공된 기본 지오코더를 사용하십시오.
- 써드 파티 벤더가 개발한 지오코더를 사용하십시오.
- 사용자 소유의 지오코더를 사용하십시오.

기본 지오코더는 주소를 지오코딩하여 이들을 ST_Point 데이터 또는 ST_Geometry 데이터 중 하나로 변환할 수 있습니다. 다른 공간 데이터 유형의 데이터를 저장해야 할 경우에는 지오코더를 사용하여 그와 같은 데이터를 생성할 수 있습니다. 주소가 없는 사이트(예: 토지 면적이 변화하는 농장)를 나타내는 공간 데이터가 필요한 경우에도, 지오코더를 plug in하여 그 요구를 충족할 수 있습니다.

plug in 지오코더를 사용하기 전에 이를 먼저 등록해야 합니다. 사용자와 벤더는 db2gse.gse_register_gc 저장 프로시듀어로 이를 등록할 수 있습니다. 제어 센터에서는 이를 등록할 수 없습니다. 97 페이지의 『db2gse.gse_register_gc』에서 db2gse.gse_register_gc에 대한 자세한 내용을 참조하십시오. 75 페이지의 『제9장 저장 프로시듀어』에서 DB2 Spatial Extender 저장 프로시듀어의 사용에 대한 일반 정보를 참조하십시오.

지오코더는 두 가지 모드로 작동합니다.

- *일괄처리 모드*에서는 공간 컬럼에 대한 기존의 모든 소스 데이터를 공간 데이터로 변환하고 해당 데이터가 있는 컬럼을 데이터 상주시키려고 하나의 조작으로 시도합니다. 지오코더 실행 창에서 이 조작을 시작할 수 있습니다. 또는, db2gse.gse_run_gc 저장 프로시듀어를 호출하도록 프로그램을 코딩하여 응용프로그램으로 이를 시작할 수 있습니다.
- *증분 모드*에서 지오코더는 컬럼 현재 값을 보유하기 위해 결과 공간 값을 위치시켜 테이블에 삽입되거나 갱신될 때 데이터를 변환합니다. 이는 공간 계층 작성 창에서 요청할 수 있는 삽입 및 갱신 트리거에 의해 활성화됩니다. 또는, db2gse.gse_enable_autogc 저장 프로시듀어를 호출하도록 프로그램을 코딩하여 응용프로그램으로 이를 요청할 수 있습니다.

  증분 지오코딩을 *자동 지오코딩*이라고도 합니다.

지오코더를 사용하려고 계획할 때는 다음과 같은 요소를 고려해야 할 것입니다.

1. 제어 센터 사용시, 지오코더 실행 창을 사용하기 전에 일반적으로 공간 계층 작성 창을 사용합니다. 이는 일괄처리 지오코딩을 시작하기 전에 DB2 Spatial Extender가 증분 지오코딩에 대한 트리거를 설정할 수 있음을 의미합니다. 그런 이유로 증분 지오코딩이 일괄처리 지오코딩에 우선할 수 있습니다. 모든 소스 데이터를 일괄처리 모드로 처리하면, 지오코더가 조작하는 동일한 데이터를 증분 모드로 지오코딩합니다. 이렇게 여유를 두면 중복이 발생되지 않습니다(공

간 데이터가 두 번 생성될 때, 데이터의 두 번째 부산물이 첫번째와 겹치게 됩니다). 그러나 이로 인해 성능이 저하될 수 있습니다. 이를 피하는 한 가지 방법은 일괄처리 지오코딩이 완료될 때까지 트리거 설정을 지연시키는 것입니다.

2. 일괄처리 모드로 지오코딩하려고 할 때 트리거가 적소에 있으면, 일괄처리 지오코딩이 종료할 때까지 트리거를 비활성화하는 것이 바람직합니다. db2gse.gse_disable_autogc 저장 프로시듀어를 호출하도록 프로그램을 코딩하여 지오코더 실행 창에서 또는 응용프로그램으로 이를 비활성화시킬 수 있습니다. 지오코더 실행 창을 사용하는 경우에는 지오코딩이 완료될 때 DB2 Spatial Extender가 자동으로 이를 재활성화합니다. db2gse.gse_disable_autogc 저장 프로시듀어를 사용하는 경우에는 db2gse.gse_enable_autogc 저장 프로시듀어를 호출하여 재활성화할 수 있습니다.

3. 색인이 있는 공간 컬럼을 데이터 상주시키기 위해 지오코더를 일괄처리 모드로 수행하려면, 색인을 먼저 사용 불가능하게 하거나 제거하십시오. 그렇지 않은 경우에, 지오코더가 실행되는 동안 색인이 조작 가능한 상태로 남아 있으면 성능이 현저하게 떨어지게 됩니다. 제어 센터를 사용할 경우에는 지오코더 실행 창에서 색인을 사용 불가능하게 할 수 있습니다. DB2 Spatial Extender는 지오코딩이 종료되면 색인을 자동으로 다시 사용 가능하게 합니다. 응용프로그램을 사용할 경우에는 SQL DROP문으로 색인을 제거할 수 있습니다. 이렇게 수행할 경우에는 일괄처리 지오코딩이 종료된 후에 색인을 다시 작성할 수 있도록 색인의 매개변수를 잘 기록해 두어야 합니다.

4. 지오코더는 소스 데이터의 레코드를 읽을 때, 해당 레코드가 참조 자료의 상대 레코드와 일치하게끔 시도합니다. 지오코더가 레코드를 처리하기 위해서는 특정 등급(*정밀도*라고 함)만큼 정확하게 일치해야 합니다. 예를 들어, 정밀도 85는 소스 레코드와 참조 자료의 상대 레코드 사이에 적어도 85%의 정확도로 일치해야 소스 레코드가 처리될 수 있음을 의미합니다.

   정밀도가 얼마인지는 사용자가 지정합니다. 이를 조정해야 할 경우도 있습니다. 예를 들어, 정밀도 100이라고 가정합시다. 많은 소스 레코드에 참조 자료 보다 더 최근의 주소가 들어 있다면, 이 레코드와 참조 자료 사이에 100%의 정확도로 일치하는 것이 불가능합니다. 결과적으로 지오코더는 이 레코드를 거부하게 됩니다. 대체적으로 지오코더가 불충분하거나 매우 부정확하게 보이는 공간 데이터를 생성하는 경우에는, 정밀도를 변경하고 지오코더를 다시 실행시켜 이 문제를 해결할 수도 있습니다.

지오코더를 일괄처리 모드로 실행

이 절에서는 제어 센터에서 일괄처리 모드로 지오코더를 수행하는 단계에 대한 개요를 제공합니다. 이 개념 뒤에 각 단계를 완성하는 방법에 대한 세부사항들이 나옵니다.

지오코더를 일괄처리 모드로 실행하는 데 필요한 권한을 알아내려면 104 페이지의 『권한』을 참조하십시오.

지오코더를 일괄처리 모드로 실행하는 단계의 개요:

1. 지오코더 실행 창을 여십시오.
2. 사용하려는 지오코더를 지정하십시오.
3. 지오코더의 성능을 방해할 수 있는 오브젝트를 사용하지 마십시오.
4. DB2가 확약을 발행하기 전에 얼마나 많은 레코드들을 지오코딩할 것인지 지정하십시오.
5. 지오코더를 어떻게 조작할 것인지 지정하십시오.
6. 지오코더를 실행하도록 DB2 Spatial Extender에 지시하십시오.

지오코더를 일괄처리 모드로 실행하는 세부 단계:

1. 지오코더 실행 창을 여십시오.
   a. 제어 센터 창에서 공간적으로 사용 가능한 데이터베이스 내의 **테이블** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오.
   b. **테이블** 폴더를 클릭하십시오. 창의 오른쪽면에 있는 목차 분할창에 테이블이 표시됩니다.
   c. 목차 분할창에서 원하는 테이블을 마우스 오른쪽 버튼으로 클릭한 후 팝업 메뉴에서 **공간 계층**을 클릭하십시오. 공간 계층 창이 열립니다.
   d. 공간 계층 창에서,
      1) 데이터를 상주시키려는 컬럼에 정의된 계층을 선택하십시오.
      2) **지오코더 실행** 누름 버튼을 클릭하십시오. 지오코더 실행 창이 열립니다.
2. 기본 지오코더를 사용하려는 경우에는 **이름** 상자의 기본 이름을 있는 그대로 두십시오. 그렇지 않으면 상자를 사용하여 원하는 지오코더를 선택하십시오.

3. 지오코더의 성능을 방해할 수 있는 오브젝트를 사용하지 마십시오.
   - 데이터를 상주시키려는 컬럼에 색인이 있으면 **지오코딩 프로세스 동안 공간 색인을 임시로 사용 안함** 선택란을 선택하십시오.
   - 이 컬럼에 대한 증분 지오코딩을 활성화하도록 트리거를 설정한 경우에는 **지오코딩 프로세스 동안 공간 색인을 임시로 사용 안함** 선택란을 선택하십시오.

     지오코더 실행 창에서 **확인**을 클릭하면 색인 및 트리거가 자동으로 재사용됩니다.
4. **확약 범위** 스핀 버튼을 사용하여 DB2가 확약을 발행하기 전에 얼마나 많은 레코드들을 지오코딩할 것인지 지정하십시오. 예를 들어, DB2가 한번에 100개의 지오코딩된 레코드를 확약하게 하려면, 숫자 100을 지정하십시오.

   팁: 모든 레코드가 처리된 후에만 DB2가 확약을 발행하게 하려면 0을 지정하십시오.
5. **지오코더 매개변수** 그룹 상자의 필드를 사용하여 지오코더를 어떻게 조작할 것인지 지정하십시오.
   - **정밀도 레벨** 스핀 버튼을 사용하여 소스 레코드와 그의 참조 자료 상대방 사이에 어느 정도 정확하게 일치해야 하는지 백분율 단위로 지정하십시오. 47 페이지의 『지오코딩 관련 정보』에서 정밀도에 대한 자세한 내용을 참조하십시오.
   - 벤더가 공급한 지오코더를 사용 중이고 그 프로그램이 지원하는 등록 등록 정보를 사용하려면, **등록 정보** 상자를 사용하여 이 등록 정보들을 설정하십시오.
   - 사용자가 선택한 테이블에 있는 행의 부분 집합만을 지오코딩하려면 **WHERE절** 상자를 사용하여 원하는 행에 대한 기준을 지정하는 SELECT WHERE절을 코드화하십시오. 이 절에서는 테이블에 있는 모든 컬럼을 참조할 수 있습니다.

     기준만을 입력하십시오. 키워드 WHERE는 생략하십시오. 예를 들어, 테이블에 STATE 컬럼이 있고 이 컬럼에 값 MA가 들어 있는 행만을 지오코딩하려는 경우에는 다음을 입력하십시오.

     ```
     STATE='MA'
     ```

6. **확인**을 클릭하여 지오코더를 실행하십시오.

데이터 가져오기 및 내보내기

이 절에서는 데이터의 가져오기 및 내보내기 프로세스에 대해 서술하고 제어 센터의 사용법에 대해 설명합니다.

- 데이터 교환 파일로부터 새 또는 기존 테이블로 데이터 가져오기
- 데이터 교환 파일로부터 기존 테이블로 데이터 가져오기
- 테이블로부터 데이터 교환 파일로 데이터 내보내기

가져오기 및 내보내기 관련 정보

이 절에서는 공간 데이터를 가져오기 및 내보내기하는 이유를 나열합니다. 또한 내보내기 소스와 가져오기 목표 사이의 인터페이스로서 처리되는 데이터 교환 파일에 대해서도 언급합니다.

DB2 Spatial Extender를 사용하여 데이터 교환 파일로부터 공간 데이터를 가져오고 그 파일로 공간 데이터를 내보내기할 수 있습니다. 다음과 같은 시나리오를 생각해 봅시다.

- GIS에는 사무소, 고객 및 기타 다른 비지니스 관련사항을 나타내는 공간 데이터가 포함됩니다. 사용자 소속의 문화적인 환경(도시, 거리, 명승지 등)을 나타내는 공간 데이터와 함께 이 데이터를 공급하려고 합니다. 사용자가 원하는 데이터는 맵 벤더에서 제공합니다. 벤더가 공급하는 데이터 교환 파일로부터 DB2 Spatial Extender를 사용하여 이를 가져오기할 수 있습니다.
- Oracle 시스템에서 DB2 Spatial Extender GIS로 공간 데이터를 이주하려고 합니다. Oracle 유틸리티를 사용하여 데이터를 데이터 교환 파일로 로드합니다. 그런 다음, DB2 Spatial Extender를 사용하여 이 파일로부터 공간 데이터 조작에 사용한 데이터베이스로 데이터를 가져오기합니다.
- GIS 브라우저를 사용하여 고객에게 공간 정보를 가시적인 표현하여 보여 주려고 합니다. 브라우저에는 작업할 파일만 필요하며, 데이터베이스에는 연결될 필요가 없습니다. DB2 Spatial Extender를 사용하여 데이터 교환 파일로 데이터를 내보내기한 후에, 브라우저 유틸리티를 사용하여 데이터를 브라우저로 로드할 수 있습니다.

제어 센터에서는 DB2 Spatial Extender에 대해 두 종류의 데이터 교환 파일 즉, 형상 파일과 ESRI_SDE 전송 파일을 지원합니다. 형상 파일은 종종 파일 시스템에서 생긴 데이터를 반입하고 파일 시스템에 로드될 파일로 데이터를 내보내기하는 데 사용됩니다. ESRI_SDE 전송 파일은 종종 ESRI 데이터베이스에서 생긴 데이터를 가져오기하는 데 사용됩니다.

새 또는 기존 테이블로 데이터 가져오기

이 절에서는 형상 또는 ESRI_SDE 전송 파일로부터 새 또는 기존 테이블로 데이터를 가져오는 단계의 개요를 제공합니다. 이 개념 뒤에 각 단계를 완성하는 방법에 대한 세부사항들이 나옵니다.

형상 데이터를 가져오기하는 데 필요한 권한을 알아내려면 95 페이지의 『권한』을 참조하십시오. ESRI_SDE 데이터를 가져오기하는 데 필요한 권한을 알아내려면 93 페이지의 『권한』을 참조하십시오.

새 또는 기존 테이블로 데이터를 가져오기 위한 단계의 개요:

1. 공간 데이터 가져오기 창을 여십시오.
2. 가져오기할 데이터가 들어 있는 파일의 경로, 이름 및 형식을 지정하십시오.
3. 각 확약 전에 얼마나 많은 레코드들을 가져오기할 것인지 지정하십시오.
4. 작성될 테이블로 공간 데이터를 가져오기하려면, 이 테이블에 대한 이름과 데이터가 예정된 컬럼에 대한 이름을 지정하십시오. 기존 테이블로 공간 데이터를 가져오기하려면, 데이터가 예정된 컬럼을 지정하십시오.
5. 데이터와 연관될 공간 참조 시스템을 지정하십시오.
6. 가져오기에 실패한 레코드들을 수집할 파일을 지정하십시오.
7. DB2 Spatial Extender에게 데이터를 가져오도록 지시한 후, 이 창에서 테이블을 정의한 경우에는 테이블을 작성하고 데이터가 예정된 컬럼을 계층으로 등록하도록 지시하십시오.

새 또는 기존 테이블로 데이터를 가져오기 위한 세부 단계:

1. 공간 데이터 가져오기 창을 여십시오.
   a. 제어 센터 창에서 DB2 Spatial Extender를 수행시킬 서버 아래의 **데이터베이스** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오.

b. **데이터베이스** 폴더를 클릭하십시오. 창의 오른쪽면에 있는 목차 분할창에 데이터베이스가 표시됩니다.

c. 데이터를 가져오기하려는 데이터베이스를 마우스 오른쪽 버튼으로 클릭한 후 팝업 메뉴에서 **Spatial Extender —> 공간 데이터 가져오기**를 클릭하십시오. 공간 데이터 가져오기 창이 열립니다.

2. 가져오기할 데이터가 들어 있는 파일의 경로, 이름 및 형식을 지정하십시오.

   a. **파일 이름** 필드를 사용하여 경로 및 이름을 지정하십시오.

   b. **파일 형식** 상자를 사용하여 형식을 지정하십시오. 형식은 다음과 같습니다.

      형상 이는 기본값입니다.

      ESRI_SDE
      이 형식을 지정하면, 이 형식과 연관된 공간 참조 시스템의 이름이 **공간 참조 이름** 필드의 기본값이 됩니다.

3. **확약 범위** 필드를 사용하여 각 확약 이전에 가져오기할 레코드수를 지정하십시오. 예를 들어, DB2가 한번에 100개의 레코드를 확약하게 하려면, 숫자 100을 지정하십시오.

   팁: 모든 레코드가 처리된 후에만 DB2가 확약을 발행하게 하려면 0을 지정하십시오.

4. 데이터가 예정된 테이블과 컬럼을 지정하십시오.

   a. **계층 스키마** 상자를 사용하여 데이터를 가져오기할 테이블에 대한 스키마를 지정하십시오.

   b. 테이블 및 컬럼을 지정하십시오.

      - 테이블이 아직 존재하지 않는 경우:

        1) **계층 테이블** 필드에 테이블에 대한 이름을 입력하십시오.

        2) **계층 컬럼** 필드에 가져오기된 데이터가 포함될 컬럼에 대한 이름을 입력하십시오. DB2 Spatial Extender는 자동으로 이 컬럼을 계층으로 등록합니다.

      - 테이블이 이미 존재하는 경우:

1) **계층 테이블** 필드에 테이블을 지정하십시오. 여기에는 가져오기된 데이터가 들어갈 컬럼이 이미 포함되어 있어야 합니다. 또한 이 컬럼이 계층으로 이미 등록된 상태이어야 합니다.
2) **계층 컬럼** 필드에 가져오기된 데이터가 예정되어 있는 컬럼의 이름을 지정하십시오.

5. **공간 참조 이름** 필드에 이 데이터와 연관될 공간 참조 시스템을 입력하거나 선택하십시오. (데이터가 ESRI_SDE 전송 파일에서 온 경우에는 관련 공간 참조 시스템의 이름이 필드에 자동으로 표시됩니다.)
6. **예외 파일** 필드에, 가져오기에 실패한 레코드들을 수집할 수 있는 새 파일의 경로와 이름을 지정하십시오. 나중에 이 레코드들을 수정하여 이 파일로부터 레코드들을 가져오기할 수 있습니다.

   DB2 Spatial Extender가 이 파일을 작성합니다. 이미 존재하는 파일은 지정하지 마십시오.
7. **확인**을 클릭하여 데이터를 가져오기하십시오. 또한, 아직 존재하지 않는 테이블에 대한 이름을 제공한 경우에는, 이 테이블이 작성되고 데이터가 예정된 컬럼을 계층으로 등록합니다. 사용자가 지정한 예외 파일도 작성됩니다.

기존 테이블로 데이터 가져오기

이 절에서는 형상 파일 또는 ESRI_SDE 전송 파일로부터 기존 테이블로 데이터를 가져오는 단계의 개요를 제공합니다. 이 개념 뒤에 각 단계를 완성하는 방법에 대한 세부사항들이 나옵니다.

형상 데이터를 가져오기하는 데 필요한 권한을 알아내려면 95 페이지의 『권한』을 참조하십시오. ESRI_SDE 데이터를 가져오기하는 데 필요한 권한을 알아내려면 93 페이지의 『권한』을 참조하십시오.

기존 테이블로 데이터를 가져오기 위한 단계의 개요:

1. 공간 데이터 가져오기 창을 여십시오.
2. 가져오기할 데이터가 들어 있는 파일의 경로와 이름을 지정하십시오.
3. 각 확약 전에 얼마나 많은 레코드들을 가져오기할 것인지 지정하십시오.
4. 가져오기할 공간 데이터가 들어 있는 컬럼을 지정하십시오.
5. 이 데이터와 연관될 공간 참조 시스템을 지정하십시오.

6. 가져오기에 실패한 레코드들을 수집할 파일을 지정하십시오.
7. DB2 Spatial Extender에게 데이터를 가져오도록 지시한 후, 아직 작성되지 않은 컬럼을 지정한 경우에는 이 컬럼을 작성하고 이를 계층으로 등록하도록 지시하십시오.

기존 테이블로 데이터를 가져오기 위한 세부 단계:

1. 공간 데이터 가져오기 창을 여십시오.
   a. 제어 센터 창에서 데이터를 가져오기하려는 데이터베이스에 대해 **테이블** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오.
   b. **테이블** 폴더를 클릭하십시오. 창의 오른쪽면에 있는 목차 분할창에 테이블이 표시됩니다.
   c. 데이터를 가져오기하려는 테이블을 마우스 오른쪽 버튼으로 클릭한 후 팝업 메뉴에서 **Spatial Extender —> 공간 데이터 가져오기**를 클릭하십시오. 공간 데이터 가져오기 창이 열립니다.
2. **파일 이름** 상자에서, 가져오기할 데이터가 들어 있는 파일의 경로와 이름을 지정하십시오.
3. **확약 범위** 상자를 사용하여 각 확약 이전에 가져오기할 레코드수를 지정하십시오. 예를 들어, DB2가 한번에 100개의 레코드를 확약하게 하려면, 숫자 100을 지정하십시오.

   팁: 모든 레코드가 처리된 후에만 DB2가 확약을 발행하게 하려면 0을 지정하십시오.
4. 가져오기할 공간 데이터가 들어 있는 컬럼을 지정하십시오.
   - 컬럼이 테이블에 아직 존재하지 않으면, **계층 컬럼** 상자를 사용하여 컬럼에 대한 이름을 입력하십시오.
   - 컬럼이 아직 존재하면, **계층 컬럼** 상자를 사용하여 컬럼의 이름을 선택하거나 입력하십시오.
5. **공간 참조 이름** 상자를 사용하여, 가져오기된 데이터와 연관될 공간 참조 시스템을 지정하십시오.
   - 테이블에 컬럼을 추가하려면 공간 참조 시스템의 이름을 입력하거나 선택하십시오.

- 가져오기된 데이터가 기존 컬럼으로 예정된 경우에는 **공간 참조 이름** 상자를 그대로 두십시오. 이는 기본 공간 참조 시스템의 이름을 표시합니다.

6. **예외 파일** 필드에, 가져오기에 실패한 레코드들을 수집할 수 있는 새 파일의 경로와 이름을 지정하십시오. 나중에 이 레코드들을 수정하여 이 파일로부터 레코드들을 가져오기할 수 있습니다.

   DB2 Spatial Extender가 이 파일을 작성합니다. 이미 존재하는 파일은 지정하지 마십시오.

7. **확인**을 클릭하여 데이터를 가져오기하십시오. 또한, 아직 존재하지 않는 컬럼을 지정한 경우에는, 이 컬럼이 작성되어 계층으로 등록됩니다. 사용자가 지정한 예외 파일도 작성됩니다.

형상 파일로 데이터 내보내기

이 절에서는 형상 파일로 데이터를 내보내는 단계의 개요를 제공합니다. 이 개념 뒤에 각 단계를 완성하는 방법에 대한 세부사항들이 나옵니다.

이 단계를 수행하는 데 필요한 권한을 알아내려면 91 페이지의 『권한』을 참조하십시오.

형상 파일로부터 데이터를 내보내기 위한 단계의 개요:

1. 공간 데이터 내보내기 창을 여십시오.
2. 내보내기될 공간 데이터가 들어 있는 컬럼을 지정하십시오.
3. 데이터 행의 부분 집합을 내보내기하려면 이 부분 집합을 DB2 Spatial Extender에 인식시키십시오.
4. 데이터를 내보내기할 파일의 경로와 이름을 지정하십시오.
5. DB2 Spatial Extender에게 데이터를 내보내기하도록 지시하십시오.

형상 파일로 데이터를 내보내기 위한 세부 단계:

1. 공간 데이터 내보내기 창을 여십시오.
   a. 제어 센터 창에서 공간 데이터가 들어 있는 데이터베이스 내의 **테이블** 또는 **뷰** 폴더를 찾을 때까지 오브젝트 트리를 펼치십시오.
   b. **테이블** 또는 **뷰** 폴더를 클릭하십시오. 창의 오른쪽면에 있는 목차 분할창에 테이블 또는 뷰가 표시됩니다.

c. 내보내기할 데이터가 들어 있는 테이블 또는 뷰를 마우스 오른쪽 버튼으로 클릭한 후 팝업 메뉴에서 **Spatial Extender —> 공간 데이터 내보내기**를 클릭하십시오. 공간 데이터 내보내기 창이 열립니다.

2. **계층 컬럼** 필드에 내보내기될 공간 데이터가 들어 있는 컬럼의 이름을 지정하십시오.
3. 테이블 행의 부분 집합을 내보내기하려면 **WHERE절** 상자를 사용하여 원하는 행에 대한 기준을 지정하는 WHERE절을 입력하십시오. 이 절에서는 데이터를 내보내기할 테이블 또는 뷰에 있는 컬럼만을 참조할 수 있습니다.

   기준만을 입력하십시오. 키워드 WHERE는 생략하십시오. 예를 들어, 테이블 또는 뷰에 STATE 컬럼이 있고 이 컬럼에 값 MA가 들어 있는 행만을 지오코딩하려는 경우에는 다음을 입력하십시오.

   ```
   STATE='MA'
   ```

4. **파일 이름** 필드에, 데이터를 가져오기할 파일의 경로와 이름을 지정하십시오.
5. **확인**을 클릭하여 데이터를 내보내기하십시오.

제6장 공간 색인 작성

이 장에서는 제어 센터를 사용하여 공간 데이터에 대한 색인을 작성하는 방법에 대해 설명합니다.

데이터가 있는 공간 컬럼을 데이터 상주시켰으면 공간 색인을 작성할 준비가 되었습니다. B-트리와 같은 일반적인 색인화 구조는 테이블 데이터에 대해 선형, 1차원 정렬을 수행합니다. 공간 데이터 조작에 사용된 테이블 데이터는 단일 항목으로 저장되지 않는 2차원입니다. 예를 들어, 다각형과 같은 공간 기하학은 하나의 공간 컬럼 또는 계층에 여러 좌표값으로 구성됩니다. B-트리색인은 공간 데이터 유형을 핸들할 수 없기 때문에 DB2 Spatial Extender가 *격자 색인*이라는 독점적인 색인화 기법을 작성했습니다. 격자 색인은 B-트리 색인을 기반으로 하며, B-트리는 2차원 데이터를 핸들하고 공간 컬럼에 대해 색인화를 수행하도록 강화되었습니다. 격자 색인은 세 계층을 지원하며 광범위 오브젝트, 크기 및 데이터 분산에 대해 뛰어난 성능을 제공하도록 설계되었습니다. 123 페이지의 『제12장 공간 색인』에서 공간 색인에 대한 자세한 내용을 참조하십시오.

공간 색인을 작성하는 데 필요한 권한을 알아내려면 86 페이지의 『권한』을 참조하십시오.

제어 센터를 사용하여 공간 색인 작성

제어 센터를 사용하여 공간 색인을 작성하려면 다음을 수행하십시오.

1. 오브젝트 트리에서 **테이블** 폴더를 선택하십시오. 기존의 모든 창이 목차 분할 창에 표시됩니다.
2. 목차 분할창에서 색인을 작성하려는 테이블을 마우스 오른쪽 버튼으로 클릭한 후 팝업 메뉴에서 **Spatial Extender —> 공간 색인**을 클릭하십시오. 공간 색인 창이 열립니다.
3. 공간 색인 창에서 **작성**을 클릭하십시오. 공간 색인 작성 창이 열립니다.
4. 작성하려는 새 공간 색인의 이름을 **이름** 필드에 입력하십시오.

주: 스키마를 지정하지 않아도 됩니다. DB2 Spatial Extender가 자동으로 스키마를 추가하여 사용자에 대한 완전한 이름을 작성합니다.

5. 색인을 작성할 계층을 **계층 컬럼** 필드에서 선택하십시오.

   계층은 DB2 Spatial Extender에 정의되거나 등록된 공간 컬럼입니다.

6. 각 필드에 지정하려는 격자 *크기* 값을 **격자 *크기*** 필드에 입력하십시오.

   셀 *크기*를 증가시켜 격자 레벨 **Finest**, **Middle** 및 **Coarsest**를 입력합니다. 이리하여 두 번째 레벨이 첫번째 셀 *크기* 보다 커야 하고 세 번째는 두 번째 보다 커야 합니다.

격자 셀 크기 판별

적절한 격자 *크기*를 결정하는 일은 시행 착오 과정을 통해 이뤄집니다. 색인화할 오브젝트의 대략적인 *크기*와 관련이 있는 격자 *크기*를 설정하는 것이 바람직합니다. 격자 *크기*를 너무 작거나 너무 *크*게 설정하면 성능이 떨어질 수 있습니다. *크기*를 너무 작게 설정하면 색인 검색 동안 키/오브젝트 비율에 영향을 미치게 됩니다. 이러한 경우에는 너무 많은 키가 작성되어 많은 수의 후보들이 리턴됩니다. 격자 *크기*를 너무 *크*게 설정하면 초기 색인 검색에서 적은 수의 후보들을 리턴하지만 최종 테이블 스캔 동안 성능이 떨어질 수도 있습니다.

130 페이지의 『격자 셀 *크기* 선택』에서 격자 셀 *크기* 및 격자 레벨 수의 선택에 대한 자세한 내용을 참조하십시오.

제7장 공간 정보 검색 및 분석

공간 색인을 구축했으면 공간 테이블을 사용할 준비가 된 것입니다. 이 장에서는 공간 데이터의 검색과 분석에 관련된 주제에 대해 토론합니다. 이에는 다양한 검색 방법에 대한 개요가 포함되며 공간 함수를 이용하는 테이블 조회의 예를 제공합니다.

공간 분석의 수행 방법

다음과 같은 프로그래밍 환경을 가진 SQL 및 공간 기능을 사용하여 공간 분석을 수행할 수 있습니다.

- geobrowser(예: ESRI의 ArcExplorer).

  *ArcExplorer 사용*에서 ArcExplorer의 사용에 대한 자세한 내용을 참조할 수 있으며, 이는 http://www.esri.com의 ESRI 웹 사이트에서 사용 가능합니다.
- 대화식 SQL문.
- 사용자 개발 응용프로그램(예: ODBC, JDBC 및 Embedded SQL).

응용프로그램은 DB2 명령 센터, DB2 명령 창 또는 명령행 처리기에서 시작할 수 있습니다.

공간 조회 빌드

이 절에서는 공간 함수와 술어를 이용하는 공간 조회의 빌드에 대해 토론합니다.

공간 함수 및 SQL

DB2 Spatial Extender에는 공간 데이터에 대해 다양한 조작을 수행하는 함수가 포함됩니다. 이 절의 예에서는 공간 함수를 사용하여 자신의 공간 조회를 빌드하는 방법을 보여 줍니다.

62 페이지의 표3에서는 공간 함수의 목록과 공간 함수가 수행하는 조작 유형을 제공합니다.

표 3. 공간 함수 및 조작

함수 유형	조작 예
계산	두 점간의 거리를 계산
비교	범람 지역에 위치한 모든 고객 찾기
데이터 교환	데이터를 지원된 형식으로 변환
변환	점에 5마일 반지름을 추가

133 페이지의 『제13장 기하학 및 관련 공간 함수』 및 173 페이지의 『제14장 SQL 조회에 대한 공간 함수』에서 공간 함수에 대한 자세한 내용을 참조하십시오.

예 1: 비교

다음 조회에서는 각 백화점으로부터의 평균 고객 거리를 알아 냅니다. 이 예에 사용된 공간 함수는 ST_Distance와 ST_Within입니다.

```
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
```

예 2: 데이터 교환

다음 조회에서는 San Francisco Bay 구역에 사는 고객의 위치를 찾아 냅니다. 이 예에 사용된 공간 함수는 ST_AsText(데이터 교환)와 ST_Within입니다. ST_AsText는 c.location 컬럼의 공간 데이터를 OGC TEXT 형식으로 변환합니다.

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

예 3: 계산

다음 조회에서는 10.5마일 보다 긴 모든 거리를 찾습니다. 이 예에 사용된 공간 함수는 ST_Length입니다.

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

예 4: 변환

이 조회에서는 범람 지역에 살거나 범람 지역의 경계에서 2마일 이내에 사는 고객을 찾습니다. 이 예에 사용된 공간 함수는 ST_Buffer(변환) 및 ST_Within입니다.

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

공간 관련 술어 및 SQL

공간 관련 술어라고 불리는 공간 함수의 특수화된 그룹이 조회 성능을 개선할 수 있습니다. 두 개의 다각형이 서로 겹치는지 알아 보기 위해 두 개의 다각형을 비교하는 ST_Overlaps와 같은 공간 관련 술어는 시간과 메모리 요구사항 양쪽 면에서 실행하는 데 부담이 클 수 있습니다. 그러므로, 실행 단가를 최소화할 수 있는 최적화 기법이 매우 중요합니다. DB2 조회 최적화 알고리즘에서는 이 절의 후반부에 서술된 규칙에 따라 공간 관련 술어를 사용할 때 공간 색인을 사용하여 조회 성능을 개선합니다. 149 페이지의 『술어 함수』에서 공간 관련 술어에 대한 자세한 내용을 참조하십시오. 공간 색인을 이용하는 데 사용된 공간 관련 술어는 다음과 같습니다.

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

173 페이지의 『제14장 SQL 조회에 대한 공간 함수』에서 모든 공간 함수와 관련 술어에 대한 전체 목록을 참조하십시오.

색인 이용 규칙

다음 규칙은 공간 관련 술어를 사용하여 공간 조회를 최적화하려는 경우에 적용됩니다.

- 술어는 WHERE절에 사용되어야 합니다.
- 술어는 비교문의 왼쪽편에 있어야 합니다. 예를 들면 다음과 같습니다.

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

- 등식 비교에서는 정수 상수 1을 사용해야 합니다.

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

- 술어에 검색 목표로 사용된 공간 컬럼이 있어야 하고 해당 컬럼에 대해 작성된 공간 색인이 있어야 합니다.

색인 이용 예

표4에서는 공간 색인을 이용하기 위해 공간 조회를 작성할 때의 올바른 방법과 틀린 방법을 보여 줍니다.

표 4. 색인 이용 규칙

공간 조회	규칙 위반
`SELECT * FROM customers c WHERE db2gse.ST_Within (c.location,:BayArea)=1`	이 예에서는 어떤 조건도 위반되지 않았습니다.
`SELECT * FROM customers c WHERE db2gse.ST_Distance (c.location,:SanJose)<10`	이 예에서는 어떤 조건도 위반되지 않았습니다.
`SELECT * FROM customers c WHERE db2gse.ST_Length (c.location)>10`	술어는 WHERE절에 사용되어야 합니다. (ST_Length는 공간 함수이지 술어가 *아닙니*.)
`SELECT * FROM customers c WHERE 1=db2gse.ST_Within (c.location,:BayArea)`	술어는 비교문의 왼쪽편에 있어야 합니다.
`SELECT * FROM customers c WHERE db2gse.ST_Within (c.location,:BayArea)=2`	등식 비교에서는 정수 상수 1을 사용해야 합니다.
`SELECT * FROM customers c WHERE db2gse.ST_Within (:SanJose,:BayArea)=1`	술어에 검색 목표로 사용된 공간 컬럼이 있어야 하고 해당 컬럼에 대해 작성된 공간 색인이 있어야 합니다. (SanJose 및 BayArea는 공간 컬럼이 아니므로 공간 컬럼과 관련된 공간 색인을 가질 수 없습니다.)

제8장 DB2 Spatial Extender용 응용프로그램 작성

이 장에서는 공간 정보에 대해 작업하고 이를 사용자 정의하기 위해 DB2 Spatial Extender 샘플 프로그램을 사용하여 응용프로그램을 작성하는 방법에 대해 설명합니다. 다음의 주제에 대해 설명합니다.

- 샘플 프로그램의 사용
- 샘플 프로그램 단계들

샘플 프로그램의 사용

DB2 Spatial Extender 샘플 프로그램을 사용하면 응용프로그램을 보다 쉽게 프로그래밍할 수 있습니다. 샘플 프로그램으로 다음을 수행할 수 있습니다.

- 일상적인 공간 프로시듀어의 자동화
- 샘플 코드를 자신의 응용프로그램으로 잘라서 붙여넣기
- 공간적으로 사용 가능한 데이터베이스를 작성하고 유지보수하는 데 일반적으로 필요한 단계를 이해함

샘플 프로그램을 사용하여 DB2 Spatial Extender에 대한 복합 타스크를 코드화하십시오. 예를 들어, DB2 Spatial Extender 저장 프로시듀어를 호출하기 위해 데이터베이스 인터페이스를 사용하는 응용프로그램을 작성하는 것입니다. 샘플 프로그램에서 응용프로그램을 복사하여 사용자 정의할 수 있습니다. DB2 Spatial Extender에 대한 프로그래밍 단계에 익숙치 않은 경우에는 각 단계를 상세히 보여 주는 샘플 프로그램을 실행할 수 있습니다. 그러나, 먼저 샘플 프로그램을 작성해야 합니다. 샘플 makefile을 이용하면 이를 수행할 수 있습니다. 샘플 프로그램의 작성 및 실행에 대한 지시사항은 23 페이지의 『설치 검증』을 참조하십시오.

샘플 프로그램 단계

표5에서는 샘플 프로그램 단계, 관련 저장 프로시듀어 및 각 단계에 대한 설명을 보여 줍니다. 저장 프로시듀어를 호출하기 위한 C 함수는 표5의 조치 컬럼에 괄호로 묶어 표시됩니다. 75 페이지의 『제9장 저장 프로시듀어』에서 저장 프로시듀어에 대한 자세한 내용을 참조하십시오. 샘플 프로그램은 14 페이지의 『시나리오 : 보험 회사에서 GIS를 갱신하는 경우』에 소개된 시나리오를 기본으로 합니다.

표 5. DB2 Spatial Extender 샘플 프로그램

샘플 프로그램 단계	조치	설명
공간 데이터베이스의 사용/사용 안함	1. 공간 데이터베이스를 사용하십시오 (gseEnableDB). 2. 공간 데이터베이스를 사용 불가능하게 하십시오(gseDisableDB). 3. 공간 데이터베이스를 사용하십시오 (gseEnableDB).	1. 이것이 DB2 Spatial Extender를 사용하는 데 필요한 첫 단계입니다. 공간 데이터 조작에 사용된 데이터베이스에는 공간 유형 세트, 공간 함수 세트, 공간 관련 술어 세트, 새로운 색인 유형 및 관리 테이블과 뷰 세트가 있습니다. 2. 이 단계는 보통 잘못된 데이터베이스에 대해 공간 기능을 사용할 때 수행됩니다. 공간 데이터베이스를 사용 안 할 때는 사용자가 공간 유형 세트, 공간 함수 세트, 공간 관련 술어 세트, 새로운 색인 유형 및 관리 테이블과 뷰 세트를 제거합니다. **주:** 데이터베이스 사용 프로시듀어에 의해 작성된 오브젝트에 종속적인 오브젝트가 작성되는 경우에는 데이터베이스 사용 안함이 실패하게 됩니다. 예를 들어, ST_Point 유형의 공간 컬럼이 있는 테이블을 작성하면 데이터베이스 사용 안함이 실패하게 됩니다. 이는 테이블이 데이터베이스 사용 안함 프로시듀어가 제거할 예정인 ST_Point 유형에 종속적이기 때문에 일어납니다. 3. 1과 같습니다.

표 5. DB2 Spatial Extender 샘플 프로그램 (계속)

샘플 프로그램 단계	조치	설명
공간 참조 시스템 등록	1. CUSTOMERS 테이블의 LOCATION 컬럼에 대해 공간 참조 시스템을 등록하십시오(gseEnableSref). 2. OFFICES 테이블의 LOCATION 컬럼에 대해 공간 참조 시스템을 등록하십시오(gseEnableSref). 3. OFFICES 테이블의 LOCATION 컬럼에 대해 공간 참조 시스템을 등록해제하십시오(gseDisableSref). 4. OFFICES 테이블의 ZONE 컬럼에 대해 공간 참조 시스템을 재등록하십시오(gseEnableSref).	1. 이 단계에서는 CUSTOMERS 테이블의 공간 데이터를 풀이하는 데 사용될 예정인 새로운 공간 참조 시스템(SRS)을 정의합니다. 공간 참조 시스템에는 공간적으로 사용 가능한 데이터베이스의 컬럼에 저장될 수 있는 양식의 기하학 데이터가 포함됩니다. SRS가 특정 계층에 등록된 후에, 해당 계층에 적용가능한 좌표들이 관련 CUSTOMERS 테이블 컬럼에 저장될 수 있습니다. 2. 이 단계에서는 OFFICES 계층의 공간 데이터를 풀이하는 데 사용될 예정인 새로운 공간 참조 시스템(SRS)을 정의합니다. 각 테이블 계층에는 그에 대해 정의된 SRS가 있어야 합니다. OFFICES 테이블 계층은 CUSTOMERS 테이블 계층보다 관련된 다른 SRS를 필요로 할 수도 있습니다. 3. 이 단계는 계층 또는 공간 컬럼에 대해 잘못된 SRS 매개변수를 지정한 경우에 수행됩니다. OFFICES 테이블 계층에 대해 SRS를 등록 해제할 때, 관련 매개변수와 함께 정의를 제거합니다. 4. 이 단계에서는 OFFICES 계층의 공간 데이터를 풀이하는 데 사용될 예정인 새로운 공간 참조 시스템(SRS)을 정의합니다.
공간 테이블 작성	1. LOCATION 컬럼(gseSetupTables)을 추가하여 CUSTOMERS 테이블을 바꾸십시오. 2. OFFICES 테이블(gseSetupTables)을 작성하십시오.	1. CUSTOMERS 테이블은 몇년 동안 데이터베이스에 저장된 비지니스 데이터를 나타냅니다. ALTER TABLE문은 ST_Point 유형의 새 컬럼(LOCATION)을 추가합니다. 이 컬럼은 다음 단계에서 주소 컬럼을 지오코딩하여 데이터 상주됩니다. 2. OFFICES 테이블은 보험 회사의 각 사무소에 대한 영업 구역을 나타냅니다. 전체 테이블은 다음 단계에서 DB2가 아닌 데이터베이스의 속성 데이터와 함께 데이터 상주됩니다. 이 단계에는 SHAPE 파일로부터 OFFICES 테이블로 속성 데이터를 가져오기하는 과정이 포함됩니다.

표 5. DB2 Spatial Extender 샘플 프로그램 (계속)

샘플 프로그램 단계	조치	설명
공간 계층 등록	1. CUSTOMERS 테이블의 LOCATION 컬럼을 계층으로 등록하십시오 (gseRegisterLayer). 2. OFFICES 테이블의 ZONE 컬럼을 계층으로 등록하십시오 (gseRegisterLayer).	이 단계에서는 LOCATION 및 ZONE 컬럼을 DB2 Spatial Extender에 대한 계층으로 등록합니다. 공간 컬럼이 DB2 Spatial Extender 유틸리티(예: 지오코더)에 의해 데이터 상주되거나 액세스되기 전에, 이를 계층으로 등록해야 합니다.
공간 계층 데이터 상주	1. CUSTOMERS 테이블의 LOCATION 컬럼에 대해 주소 데이터를 지오코딩하십시오(gseRunGC). 2. 추가 모드를 사용하여 OFFICES 테이블을 로드하십시오(gseImportShape). 3. 작성 모드를 사용하여 HAZARD_ZONE 테이블을 로드하십시오(gseImportShape).	1. 이 단계에서는 지오코더 유틸리티를 호출하여 일괄처리 지오코딩을 수행합니다. 일괄처리 지오코딩은 테이블의 중요 부분이 지오코딩 또는 재지오코딩되어야 할 때 보통 수행됩니다. 2. 이 단계에서는 SHAPE 파일 양식의 기존 공간 데이터가 있는 OFFICES 테이블을 로드합니다. OFFICES 테이블이 존재하고 OFFICES/ZONE 계층이 등록된 상태이기 때문에, 로드 유틸리티는 기존 테이블에 새 레코드를 추가합니다. 3. 이 단계에서는 SHAPE 파일 양식의 기존 공간 데이터가 있는 HAZARD_ZONE 계층을 로드합니다. 테이블과 계층이 존재하지 않기 때문에, 로드 유틸리티는 데이터가 로드되기 전에 테이블을 작성하고 계층을 등록합니다.

표 5. DB2 Spatial Extender 샘플 프로그램 (계속)

샘플 프로그램 단계	조치	설명
공간 색인 사용	1. CUSTOMERS 테이블의 LOCATION 컬럼에 대해 공간 색인을 사용하십시오(gseEnableIdx). 2. OFFICES 테이블의 ZONE 컬럼에 대해 공간 색인을 사용하십시오(gseEnableIdx). 3. OFFICES 테이블의 LOCATION 컬럼에 대해 공간 색인을 사용하십시오(gseEnableIdx). 4. HAZARD_ZONE 테이블의 BOUNDRY 컬럼에 대해 공간 색인을 사용하십시오(gseEnableIdx).	이 단계에서는 CUSTOMERS, OFFICES 및 HAZARD_ZONE 테이블에 대해 공간 색인을 사용합니다.
자동 지오코딩 사용	1. CUSTOMERS 테이블의 LOCATION 및 ADDRESS 컬럼에 대해 자동 지오코딩을 사용하십시오(gseEnableAutoGC).	이 단계에서는 지오코더의 자동 호출을 설정합니다. 자동 지오코딩을 사용하면 CUSTOMERS 테이블의 LOCATION 및 ADDRESS 컬럼이 후속 삽입 및 갱신 조작에 대해 서로 동기화됩니다.
CUSTOMERS 테이블 삽입/갱신	1. 다른 거리의 일부 레코드를 삽입하십시오(gseInsDelUpd). 2. 일부 레코드를 새로운 주소로 갱신하십시오(gseInsDelUpd).	이 단계에서는 CUSTOMERS 테이블의 LOCATION 컬럼에 대해 삽입 및 갱신을 데모합니다. 일단 자동 지오코딩이 사용되면, ADDRESS 컬럼으로부터의 정보가 LOCATION 컬럼에 삽입되거나 갱신될 때 자동으로 지오코딩됩니다. 이 프로세스는 이전 단계에서 사용되었습니다.
자동 지오코딩 사용 안 함	1. CUSTOMERS 계층에 대해 자동 지오코딩을 사용하지 마십시오(gseDisableAutoGC). 2. CUSTOMERS 계층에 대해 공간 색인을 사용하지 마십시오(gseDisableIdxCustomersLayer).	이 단계에서는 다음 단계에 대한 준비 과정으로 지오코더와 공간 색인의 자동 호출을 사용 불가능하게 합니다(다음 단계에는 전체 CUSTOMERS 테이블의 재지오코딩 과정이 포함됨)). 많은 양의 geodata를 로드할 경우에는 데이터를 로드하기 전에 공간 색인을 사용 불가능하게 한 후 로드가 완료되면 이를 사용 가능하게 하십시오.

표 5. DB2 Spatial Extender 샘플 프로그램 (계속)

샘플 프로그램 단계	조치	설명
CUSTOMERS 테이블을 재지오코딩	1. 보다 낮은 정밀도 레벨(100% 대신 90%)로 CUSTOMERS 계층을 다시 지오코딩하십시오(gseRunGC). 2. CUSTOMERS 계층에 대해 공간 색인을 다시 사용 가능하게 하십시오(gseEnableIdx). 3. 보다 낮은 정밀도 레벨(100% 대신 90%)로 자동 지오코딩을 다시 사용 가능하게 하십시오(gseEnableAutoGC).	이 단계에서는 지오코더를 일괄처리 모드로 다시 실행하고, 새로운 정밀도 레벨로 자동 지오코딩을 다시 사용 가능하게 하며, 공간 색인과 자동 지오코딩도 다시 사용 가능하게 합니다. 이러한 조치는 공간 관리자가 지오코딩 프로세스에서 높은 실패율을 경고할 때 효과적입니다. 정밀도 레벨을 100%로 설정하면, 참조 자료에서 일치하는 주소를 찾을 수 없기 때문에 주소를 지오코딩하는 데 실패할 수도 있습니다. 정밀도 레벨를 줄이면 지오코더가 일치하는 데이터를 찾을 수 있는 기회가 더 많아집니다. 테이블이 일괄처리 모드로 재지오코딩된 후, 후속 삽입 및 갱신에 대해 공간 색인과 공간 컬럼의 유지보수를 점차적으로 촉진하도록 자동 지오코딩 및 공간 색인 양쪽 모두가 다시 사용됩니다.
뷰를 작성하고 공간 컬럼을 뷰 계층으로 등록	1. CUSTOMERS 테이블과 HAZARD_ZONE 테이블의 결합에 기초하여 HIGHRISK_CUSTOMERS 뷰를 작성하십시오(gseCreateView). 2. 뷰의 공간 컬럼을 뷰 계층으로 등록하십시오(gseRegisterLayer).	이 단계에서는 뷰를 작성하고 공간 컬럼을 뷰 계층으로 등록합니다.

표 5. DB2 Spatial Extender 샘플 프로그램 (계속)

샘플 프로그램 단계	조치	설명
공간 분석 수행	1. 각 사무소의 평균 고객 간격을 찾으십시오(ST_Within, ST_Distance). 2. 각 사무소에 대한 평균 고객 수입 및 보험료를 찾으십시오(ST_Within). 3. 기존 사무소가 담당하지 않은 고객을 찾으십시오(ST_Within). 4. 각 사무소 구역이 겹치는 위험 구역수를 찾으십시오(ST_Overlaps). 5. 사무소가 사무소 구역의 중앙에 위치한다고 가정하고 특정 고객 위치로부터 가장 가까운 사무소를 찾으십시오(ST_Distance, ST_Centroid). 6. 특정 위험 구역의 경계에 가까운 위치에 있는 고객을 찾으십시오(ST_Buffer, ST_Overlaps). 7. 특정 사무소에서 담당하는 위험도가 높은 고객을 찾으십시오. (모든 단계에서 gseRunSpatialQueries를 이용합니다)	이 단계에서는 DB2 SQL 언어로 된 공간 관련 술어와 함수를 사용하여 공간 분석을 수행합니다. DB2 조회 최적화 알고리즘은 가능할 때마다 공간 컬럼의 공간 색인을 이용하여 조회 성능을 개선합니다.
파일로 공간 계층 내보내기	highRiskCustomers 계층을 내보내십시오(gseExportShape).	이 단계에서는 조회 결과를 SHAPE 파일로 내보내기하는 예를 보여 줍니다. 다른 파일 형식으로 조회 결과를 내보내기하면 써드 파티 도구(예: ESRI ArcInfo)에서 그 정보를 사용할 수 있습니다.

제2부 참조 자료

제9장 저장 프로시듀어

이 장에서는 DB2 Spatial Extender로 지리적 정보 시스템을 빌드할 수 있는 저장 프로시듀어에 대해 설명합니다. DB2 Spatial Extender를 사용 가능하게 하여 제어 센터에서 사용할 때는 이 저장 프로시듀어를 내재적으로 호출해야 합니다. 예를 들어, DB2 Spatial Extender 창에서 **확인**을 호출할 때 DB2는 저장 프로시듀어 또는 사용자에 대한 해당 창과 연관된 저장 프로시듀어를 호출합니다. 또는 응용프로그램에서 저장 프로시듀어를 명확하게 호출할 수 있습니다. 그러한 프로그램에서는 머리글 파일 db2gse.h를 포함하는 것이 바람직합니다. 이 파일에는 사용자가 저장 프로시듀어의 매개변수에 지정하는 상수에 대한 매크로 정의가 포함됩니다. AIX에서는 이 파일이 $DB2INSTANCE/sqllib/include/ 디렉토리에 저장됩니다. Windows NT에서는 %DB2PATH%\include\ 디렉토리에 저장됩니다.

주의:

저장 프로시듀어의 입력 매개변수에 대한 모든 문자열 상수는 대소문자를 구별합니다. 이러한 상수를 필요로 하는 매개변수를 찾으려면 이 장의 테이블을 참조하십시오.

저장 프로시듀어를 내재적으로 또는 명확하게 호출하기 전에 DB2 Spatial Extender가 설치된 데이터베이스에 사용자가 연결되어 있어야 합니다. 사용자가 사용하는 첫번째 저장 프로시듀어는 db2gse.gse_enable_db입니다. 이는 공간 데이터 조작용으로 데이터베이스를 사용할 수 있습니다. 다른 저장 프로시듀어는 데이터베이스가 사용 가능해진 뒤에만 사용할 수 있습니다.

저장 프로시듀어의 구현은 DB2 Spatial Extender 서버의 db2gse 라이브러리에 아카이브됩니다.

다음 목록을 사용하여 저장 프로시듀어가 수행할 이름이나 타스크로 저장 프로시듀어를 탐색할 수 있습니다. 첫번째 목록에는 이름이 나열됩니다.

- 78 페이지의『db2gse.gse_disable_autogc』
- 80 페이지의『db2gse.gse_disable_db』
- 81 페이지의『db2gse.gse_disable_sref』
- 82 페이지의『db2gse.gse_enable_autogc』
- 85 페이지의『db2gse.gse_enable_db』
- 86 페이지의『db2gse.gse_enable_idx』
- 88 페이지의『db2gse.gse_enable_sref』
- 91 페이지의『db2gse.gse_export_shape』
- 93 페이지의『db2gse.gse_import_sde』
- 95 페이지의『db2gse.gse_import_shape』
- 97 페이지의『db2gse.gse_register_gc』
- 99 페이지의『db2gse.gse_register_layer』
- 104 페이지의『db2gse.gse_run_gc』
- 106 페이지의『db2gse.gse_unregist_gc』
- 107 페이지의『db2gse.gse_unregist_layer』

다음 목록에서는 저장 프로시듀어가 수행하는 타스크들을 나열합니다.

- 공간 컬럼에 대한 색인을 작성함(86 페이지의『db2gse.gse_enable_idx』참조).
- 공간 참조 시스템을 작성함(공간 참조 시스템 참조).88 페이지의『db2gse.gse_enable_sref』).
- 지오코더를 사용 불가능하게 하여 공간 컬럼을 해당 속성 컬럼으로 자동으로 동기화할 수 없도록 함(78 페이지의『db2gse.gse_disable_autogc』참조).
- 데이터베이스에서 공간 데이터 조작에 대한 지원을 사용하지 않음(80 페이지의『db2gse.gse_disable_db』참조).
- 공간 참조 시스템을 제거함(81 페이지의『db2gse.gse_disable_sref』참조).
- 공간 데이터 조작을 지원하도록 데이터베이스를 사용함(85 페이지의『db2gse.gse_enable_db』참조).
- 지오코더를 사용하여 공간 컬럼을 해당 속성 컬럼으로 자동으로 동기화시킴(82 페이지의『db2gse.gse_enable_autogc』참조).

- 계층 및 관련 테이블을 형상 파일로 내보내기(91 페이지의 『db2gse.gse_export_shape』 참조).
- 계층 및 관련 테이블을 ESRI_SDE 전송 파일로부터 가져오기(93 페이지의 『db2gse.gse_import_sde』 참조).
- 계층 및 관련 테이블을 형상 파일로부터 가져오기(95 페이지의 『db2gse.gse_import_shape』 참조).
- 기본 지오코더 이외의 다른 지오코더를 등록(97 페이지의 『db2gse.gse_register_gc』 참조).
- 공간 컬럼을 계층으로 등록(99 페이지의 『db2gse.gse_register_layer』 참조).
- 지오코더를 일괄처리 모드로 실행(106 페이지의 『db2gse.gse_unregist_gc』 참조).
- 기본 지오코더 이외의 다른 지오코더를 등록 해제(107 페이지의 『db2gse.gse_unregist_layer』 참조).
- 계층 등록 해제(107 페이지의 『db2gse.gse_unregist_layer』 참조).

3 페이지의 『제1장 DB2 Spatial Extender 관련 정보』 및 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이 타스크의 수행 순서에 대한 자세한 내용을 참조하십시오.

db2gse.gse_disable_autogc

이 저장 프로시듀어를 사용하여 관련 속성 컬럼(들)로 동기화된 공간 컬럼을 보존하는 트리거를 제거하거나 임시로 사용하지 마십시오. 예를 들어, 속성 컬럼(들)의 값을 일괄처리 모드로 지오코딩하는 동안에는 트리거를 사용하지 않는 것이 바람직합니다. 47 페이지의 『지오코딩 관련 정보』에서 이에 대한 자세한 내용을 참조하십시오.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseDisableAutoGc를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender 용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어가 호출되는 사용자 ID는 권한, 특권 또는 특권 세트의 양식으로 권한을 보유해야 합니다.

- 제거되거나 임시로 사용하지 않을 트리거가 정의된 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한.
- 이 테이블에서의 CONTROL 특권.
- 이 테이블에서의 ALTER, SELECT 및 UPDATE 특권.

입력 매개변수

표 6. db2gse.gse_disable_autogc 저장 프로시듀어에 대한 입력 매개변수

이름	데이터 유형	설명
operMode	SMALLINT	트리거를 제거하거나 임시로 사용하지 않을 것인지 여부를 지정합니다. 이 매개변수는 널 입력이 가능하지 않습니다. **주석:** 트리거를 제거하려면 GSE_AUTOGC_DROP 매크로를 사용하십시오. 이들을 임시로 사용하지 않으려면 GSE_AUTOGC_INVALIDATE 매크로를 사용하십시오. 이 매크로와 연관된 값을 알아내려면 db2gse.h 파일을 참조하십시오. AIX에서는 이 파일이 $DB2INSTANCE/sqllib/include/ 디렉토리에 저장됩니다. Windows NT에서는 %DB2PATH%\include\ 디렉토리에 저장됩니다.

표 6. db2gse.gse_disable_autogc 저장 프로시듀어에 대한 입력 매개변수 (계속)

이름	데이터 유형	설명
layerSchema	VARCHAR(30)	layerTable 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수는 널 입력이 가능합니다. **주석:** layerSchema 매개변수에 대한 값을 제공하지 않은 경우에는 db2gse.gse_disable_autogc 저장 프로시듀어가 호출된 사용자 ID를 기본값으로 합니다.
layerTable	VARCHAR(128)	제거 또는 임시로 사용하지 않으려는 트리거가 정의된 테이블의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
layerColumn	VARCHAR(128)	제거 또는 임시로 사용하지 않으려는 트리거에 의해 유지보수되는 공간적으로 사용 가능한 컬럼의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.

출력 매개변수

표 7. db2gse.gse_disable_autogc 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_disable_db

이 저장 프로시듀어를 사용하여 DB2 Spatial Extender가 공간 데이터를 저장하고 이 데이터에 대해 수행된 조작들을 지원할 수 있도록 자원을 제거하십시오.

이 저장 프로시듀어의 목적은 공간 데이터 조작에 대해 데이터베이스를 사용할 수 있게 된 후 데이터베이스에 공간 테이블 컬럼이나 데이터를 추가하기 전에 발생되는 문제점이나 쟁점 사항들을 해결하는 데 도움을 주는 것입니다. 예를 들어, 공간 데이터 조작에 대해 데이터베이스를 사용 가능하게 한 후, DB2 Spatial Extender를 다른 데이터베이스에 대신 사용하기로 결정한 경우입니다. 공간 컬럼 또는 가져오기된 공간 데이터를 정의하지 않은 한, 이 저장 프로시듀어를 호출하여 첫번째 데이터베이스로부터의 모든 공간 자원을 제거할 수 있습니다.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseDisableDB를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 DB2 Spatial Extender 자원을 제거할 데이터베이스에 대해 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

출력 매개변수

표 8. db2gse.gse_disable_db 저장 프로시듀어에 대한 출력 매개변수

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_disable_sref

이 저장 프로시듀어를 사용하여 공간 참조 시스템을 제거하십시오. 이 저장 프로시듀어가 처리될 때, 공간 참조 시스템에 대한 정보가 DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰에서 제거됩니다. 121 페이지의 『DB2GSE.SPATIAL_REF_SYS』에서 이 뷰에 대한 자세한 내용을 참조하십시오.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseDisableSref를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

아무 것도 필요하지 않습니다.

입력 매개변수

표 9. db2gse.gse_disable_sref 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
srId	INTEGER	제거될 공간 참조 시스템의 숫자 식별자. 이 매개변수는 널 입력이 가능하지 않습니다.

출력 매개변수

표 10. db2gse.gse_disable_sref 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

제한사항

공간 참조 시스템을 제거하기 전에 공간 참조 시스템을 사용하는 모든 계층을 등록 해제해야 합니다. 이러한 계층을 등록 해제하지 않으면, 공간 참조 시스템을 제거하기 위한 요청이 거부됩니다.

db2gse.gse_enable_autogc

이 저장 프로시듀어를 사용하여 다음을 수행할 수 있습니다.

- 관련 속성 컬럼(들)로 동기화된 공간 컬럼을 보존하는 트리거를 작성하십시오. 값이 속성 컬럼(들)에 삽입되거나 갱신될 때마다, 트리거는 등록된 상태의 지오코더를 호출하여 삽입 또는 갱신된 값을 지오코딩하고 그 결과 데이터를 공간 컬럼에 위치시킵니다.
- 임시로 사용 불가능하게 된 이후에 트리거를 다시 활성화시키십시오.
- 삽입 및 갱신된 값을 지오코딩하는 데 사용할 함수를 설정하십시오.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseEnableAutoGC를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender 용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어가 호출되는 사용자 ID는 권한, 특권 또는 특권 세트의 양식으로 권한을 보유해야 합니다.

- 이 저장 프로시듀어가 작성한 트리거가 정의되는 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한.
- 이 테이블에서의 CONTROL 특권.
- 이 테이블에서의 ALTER, SELECT 및 UPDATE 특권.

입력 매개변수

표 11. db2gse.gse_enable_autogc 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
operMode	SMALLINT	지오코딩을 시작하는 트리거가 맨처음 작성될 것인지 또는 임시로 사용 불가능해진 뒤에 재활성화될 것인지 여부를 지정하는 값. 이 매개변수는 널 입력이 가능하지 않습니다. **주석:** 트리거를 작성하려면 GSE_AUTOGC_CREATE 매크로를 사용하십시오. 이들을 재활성화하려면 GSE_AUTOGC_RECREATE 매크로를 사용하십시오. 이 매크로와 연관된 값을 알아내려면 db2gse.h 파일을 참조하십시오. AIX에서는 이 파일이 $DB2INSTANCE/sqllib/include/ 디렉토리에 저장됩니다. Windows NT에서는 %DB2PATH%\include\ 디렉토리에 저장됩니다.
layerSchema	VARCHAR(30)	layerTable 매개변수에 지정된 테이블이 속해 있는 스키마의 이름. 이 매개변수는 널 입력이 가능합니다. **주석:** layerSchema 매개변수에 대한 값을 제공하지 않은 경우에는 db2gse.gse_enable_autogc 저장 프로시듀어가 호출된 사용자 ID를 기본값으로 합니다.
layerTable	VARCHAR(128)	이 저장 프로시듀어에 의해 재활성화되거나 작성된 트리거가 운용될 테이블의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
layerColumn	VARCHAR(128)	이 저장 프로시듀어가 작성하거나 재활성화하는 트리거에 의해 유지보수될 공간 컬럼의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
gcId	INTEGER	이 저장 프로시듀어가 작성하거나 재활성화하는 삽입 및 갱신 트리거에 의해 호출될 지오코더의 식별자. operMode 매개변수가 GSE_AUTOGC_CREATE로 설정되면, 이 매개변수는 널 입력이 가능하지 않습니다. 이는 operMode가 GSE_AUTOGC_RECREATE로 설정되는 경우에 널 입력이 가능합니다.

표 11. db2gse.gse_enable_autogc 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
precisionLevel	INTEGER	지오코더가 소스 데이터를 정상적으로 처리하기 위해 소스 데이터가 해당 참조 자료와 일치해야 하는 정도. operMode 매개변수가 GSE_AUTOGC_CREATE로 설정되면, 이 매개변수는 널 입력이 가능하지 않습니다. 이는 operMode가 GSE_AUTOGC_RECREATE로 설정되는 경우에 널 입력이 가능합니다. **주석:** 정밀도 레벨의 범위는 1 - 100%입니다.
vendorSpecific	VARCHAR(256)	벤더가 제공하는 기술 정보. 예를 들어, 벤더가 매개변수를 설정하는 데 사용하는 파일의 경로와 이름. operMode 매개변수가 GSE_AUTOGC_CREATE로 설정되면, 이 매개변수는 널 입력이 가능하지 않습니다. 이는 operMode가 GSE_AUTOGC_RECREATE로 설정되는 경우에 널 입력이 가능합니다.

출력 매개변수

표 12. db2gse.gse_enable_autogc 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

제한사항

- layerColumn 매개변수는 테이블 계층으로 등록된 컬럼을 참조해야 합니다.
- operMode 매개변수가 GSE_AUTOGC_CREATE로 설정되면, 등록된 지오코더의 식별자를 gcId 매개변수로 지정해야 합니다.

db2gse.gse_enable_db

이 저장 프로시듀어를 사용하여 공간 데이터를 저장하고 조작들을 지원하는 데 필요한 자원을 데이터베이스에 제공하십시오. 이 자원에는 공간 데이터 유형, 공간 색인 유형, 카탈로그 테이블 및 뷰, 제공된 함수 및 다른 저장 프로시듀어가 포함됩니다.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseEnableDB를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 사용될 데이터베이스에 대해 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

출력 매개변수

표 13. db2gse.gse_enable_db 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_enable_idx

이 저장 프로시듀어를 사용하여 공간 컬럼에 대한 색인을 작성하십시오.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseEnableIdx를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 사용 가능해진 색인이 사용될 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한.
- 이 테이블에 대한 CONTROL 또는 INDEX 특권.

입력 매개변수

표 14. db2gse.gse_enable_idx 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
layerSchema	VARCHAR(30)	layerTable 매개변수에 지정된 테이블이 속해 있는 스키마의 이름. 이 매개변수는 널 입력이 가능합니다. **주석:** layerSchema 매개변수에 대한 값을 제공하지 않은 경우에는 db2gse.gse_enable_idx 저장 프로시듀어가 호출된 사용자 ID를 기본값으로 합니다.
layerTable	VARCHAR(128)	작성할 색인이 정의될 테이블의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
layerColumn	VARCHAR(128)	작성할 색인의 도움을 받아 탐색될 공간적으로 사용 가능한 컬럼의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.

표 14. db2gse.gse_enable_idx 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
indexName	VARCHAR(128)	작성될 색인의 이름. 이 매개변수는 널 입력이 가능하지 않습니다. **주석:** 스키마 이름을 지정하지 마십시오. DB2 Spatial Extender는 layerSchema 매개변수가 참조하는 스키마에 색인을 자동으로 지정합니다.
gridSize1	DOUBLE	미세한 색인 격자의 조절 간격을 지정하는 숫자. 이 매개변수는 널 입력이 가능하지 않습니다.
gridSize2	DOUBLE	(1) 이 색인에 대해 두 번째 격자가 없을 것임을 또는 (2) 두 번째 격자의 조절 간격을 나타내는 숫자. 이 매개변수는 널 입력이 가능합니다. **주석:** 두 번째 격자가 없을 경우에는 0을 지정하십시오. 두 번째 격자를 원한다면 이 값이 gridSize1이 지정한 격자보다 조절 간격이 더 적어야 합니다.
gridSize3	DOUBLE	(1) 이 색인에 대해 세 번째 격자가 없을 것임을 또는 (2) 세 번째 격자의 조절 간격을 나타내는 숫자. 이 매개변수는 널 입력이 가능합니다. **주석:** 세 번째 격자가 없을 경우에는 0을 지정하십시오. 세 번째 격자를 원한다면 이 값이 gridSize2가 지정한 격자보다 조절 간격이 더 적어야 합니다.

출력 매개변수

표 15. db2gse.gse_enable_idx 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_enable_sref

이 저장 프로시듀어를 사용하여, 특정 좌표 시스템의 음수 및 십진수를 DB2 Spatial Extender가 저장할 수 있도록 양수로 변환하는 방법을 지정하십시오. 사용자의 스펙을 모두 묶어서 *공간 참조 시스템*이라고 합니다. 이 저장 프로시듀어가 처리될 때, 공간 참조 시스템에 대한 정보가 DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰에 추가됩니다. 121 페이지의 『DB2GSE.SPATIAL_REF_SYS』에서 이 뷰에 대한 자세한 내용을 참조하십시오.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseEnableSref를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

아무 것도 필요하지 않습니다.

입력 매개변수

표 16. db2gse.gse_enable_sref 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
srId	INTEGER	공간 참조 시스템에 대한 숫자 식별자. 이 매개변수는 널 입력이 가능하지 않습니다. **주석:** 이 식별자는 공간적으로 사용 가능한 데이터베이스 내에서 고유해야 합니다.
srName	VARCHAR(64)	공간 참조 시스템에 대한 간단한 설명. 이 매개변수는 널 입력이 가능하지 않습니다. **주석:** 이 설명은 공간적으로 사용 가능한 데이터베이스 내에서 고유해야 합니다.
falsex	DOUBLE	음의 X 좌표값에서 뺄 때 음수가 아닌 숫자(즉, 양수 또는 0)가 남는 숫자. 이 매개변수는 널 입력이 가능하지 않습니다.

표 16. db2gse.gse_enable_sref 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
falsey	DOUBLE	음의 Y 좌표값에서 뺄 때 음수가 아닌 숫자(즉, 양수 또는 0)가 남는 숫자. 이 매개변수는 널 입력이 가능하지 않습니다.
xyunits	DOUBLE	10진 X 좌표 또는 10진 Y 좌표로 곱할 때 32비트 데이터 항목으로 저장될 수 있는 정수를 산출하는 숫자. 이 매개변수는 널 입력이 가능하지 않습니다.
falsez	DOUBLE	음의 Z 좌표값에서 뺄 때 음수가 아닌 숫자(즉, 양수 또는 0)가 남는 숫자. 이 매개변수는 널 입력이 가능하지 않습니다.
zunits	DOUBLE	10진 Z 좌표로 곱할 때 32비트 데이터 항목으로 저장될 수 있는 정수를 산출하는 숫자. 이 매개변수는 널 입력이 가능하지 않습니다.
falsem	DOUBLE	음의 치수에서 뺄 때 음수가 아닌 숫자(즉, 양수 또는 0)가 남는 숫자. 이 매개변수는 널 입력이 가능하지 않습니다.
munits	DOUBLE	10진 치수로 곱할 때 32비트 데이터 항목으로 저장될 수 있는 정수를 산출하는 숫자. 이 매개변수는 널 입력이 가능하지 않습니다.
scId	INTEGER	공간 참조 시스템이 파생될 좌표 시스템의 숫자 식별자. DB2GSE.COORD_REF_SYS 카탈로그 뷰 119 페이지의 『DB2GSE.COORD_REF_SYS』에서 좌표 시스템의 숫자 식별자가 무엇인지 알아 내십시오. 이 매개변수는 널 입력이 가능하지 않습니다.

출력 매개변수

표 17. db2gse.gse_enable_sref 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_export_shape

이 저장 프로시듀어를 사용하여 계층 및 그의 관련 테이블을 형상 파일로 내보내기하거나 새 형상 파일을 작성하여 계층 및 그의 관련 테이블을 이 새 파일로 내보내기하십시오.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseExportShape를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 내보내기될 테이블에 대해 SELECT 특권을 보유하고 있어야 합니다.

입력 매개변수

표 18. db2gse.gse_export_shape 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
layerSchema	VARCHAR(30)	layerTable 매개변수에 지정된 테이블이 속해 있는 스키마의 이름. 이 매개변수는 널 입력이 가능합니다. **주석:** layerSchema 매개변수에 대한 값을 제공하지 않은 경우에는 db2gse.gse_export_shape 저장 프로시듀어가 호출된 사용자 ID를 기본값으로 합니다.
layerTable	VARCHAR(128)	내보내기할 테이블의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
layerColumn	VARCHAR(30)	내보내기할 계층으로 등록된 컬럼의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
fileName	VARCHAR(128)	지정된 계층이 내보내기될 형상 파일의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.

표 18. db2gse.gse_export_shape 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
whereClause	VARCHAR(1024)	SQL WHERE절의 본체. 이는 지오코딩될 레코드 세트에 대한 제한사항을 정의합니다. 절에서는 내보내기할 테이블에 있는 모든 속성 컬럼을 참조할 수 있습니다. 이 매개변수는 널 입력이 가능합니다.

출력 매개변수

표 19. db2gse.gse_export_shape 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

제한사항

한번에 하나의 계층만 내보내기할 수 있습니다.

db2gse.gse_import_sde

이 저장 프로시듀어를 사용하여 SDE 전송 파일을 공간 데이터 조작에 사용된 데이터베이스로 가져오기하십시오. 저장 프로시듀어는 다음 두 가지 방법으로 작동할 수 있습니다.

- 등록된 계층 컬럼이 있는 기존 테이블의 목표가 SDE 전송 파일인 경우에는, DB2 Spatial Extender가 파일 데이터가 있는 테이블을 로드합니다.
- 그렇지 않은 경우에는 DB2 Spatial Extender가 공간 컬럼이 있는 테이블을 작성하고 이 컬럼을 계층으로 등록하며 계층과 파일 데이터가 들어 있는 테이블의 다른 컬럼들을 로드합니다.

SDE 전송 파일에 지정된 공간 참조 시스템은 DB2 Spatial Extender로 등록된 공간 참조 시스템과 비교됩니다. 지정된 시스템이 등록된 시스템과 일치하면, 로드될 때 등록된 시스템이 규정한 방법으로 전송 데이터의 음수 및 10진 값이 수정됩니다. 지정된 시스템이 등록된 시스템 중 어느 것과도 일치하지 않으면, DB2 Spatial Extender가 새 공간 참조 시스템을 작성하여 수정사항을 규정합니다.

권한

기존 테이블로 데이터를 가져오기할 때, 이 저장 프로시듀어를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 데이터가 가져오기될 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한.
- 이 테이블에서의 CONTROL 특권.

데이터를 가져오기하려는 테이블을 작성해야 할 때, 이 저장 프로시듀어를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 작성될 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한.

입력 매개변수

표 20. db2gse.gse_import_sde 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
layerSchema	VARCHAR(30)	layerTable 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수는 널 입력이 가능합니다. **주석:** layerSchema 매개변수에 대한 값을 제공하지 않은 경우에는 db2gse.gse_import_sde 저장 프로시듀어가 호출된 사용자 ID를 기본값으로 합니다.
layerTable	VARCHAR(128)	SDE 전송 파일이 로드될 테이블의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
layerColumn	VARCHAR(30)	SDE 전송 파일의 공간 데이터가 로드될 계층으로 등록된 컬럼의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
fileName	VARCHAR(128)	가져오기될 SDE 전송 파일의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
commitScope	INTEGER	체크점당 레코드 수. 이 매개변수는 널 입력이 가능합니다.

출력 매개변수

표 21. db2gse.gse_import_sde 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_import_shape

이 저장 프로시듀어를 사용하여 형상 파일을 공간 데이터 조작에 사용된 데이터베이스로 가져오기하십시오. 저장 프로시듀어는 다음 두 가지 방법으로 작동할 수 있습니다.

- 등록된 계층 컬럼이 있는 기존 테이블의 목표가 형상 파일인 경우에는, DB2 Spatial Extender가 파일 데이터가 있는 테이블을 로드합니다.
- 그렇지 않은 경우에는 DB2 Spatial Extender가 공간 컬럼이 있는 테이블을 작성하고 이 컬럼을 계층으로 등록하며 계층과 파일 데이터가 들어 있는 테이블의 다른 컬럼들을 로드합니다.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseImportShape를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 가져오기된 형상 데이터가 로드될 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한.
- 이 테이블에서의 CONTROL 특권.

입력 매개변수

표 22. db2gse.gse_import_shape 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
layerSchema	VARCHAR(30)	layerTable 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수는 널 입력이 가능합니다. **주석:** layerSchema 매개변수에 대한 값을 제공하지 않은 경우에는 db2gse.gse_import_shape 저장 프로시듀어가 호출된 사용자 ID를 기본값으로 합니다.

표 22. db2gse.gse_import_shape 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
layerTable	VARCHAR(128)	가져오기된 형상 파일이 로드될 테이블의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
layerColumn	VARCHAR(30)	형상 파일이 로드될 계층으로 등록된 컬럼의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
fileName	VARCHAR(128)	가져오기될 형상 파일의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
exceptionFile	VARCHAR(128)	가져오기될 수 없는 형상이 저장된 파일의 경로와 이름. 이는 db2gse.gse_import_shape 저장 프로시듀어가 실행될 때 작성되는 새 파일입니다. 이 매개변수는 널 입력이 가능하지 않습니다.
srId	INTEGER	형상 파일이 로드될 계층에 대해 사용될 공간 참조 시스템의 식별자. 이 매개변수는 널 입력이 가능합니다. **주석:** 이 식별자가 지정되지 않은 경우에는 형상 파일에 대해 가능한 최대 해상도로 내부 변환이 설정됩니다.
commitScope	INTEGER	체크점당 레코드 수. 이 매개변수는 널 입력이 가능합니다.

출력 매개변수

표 23. db2gse.gse_import_shape 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_register_gc

이 저장 프로시듀어를 사용하여 기본값 이외의 다른 지오코더를 등록하십시오. 지오코더가 이미 등록된 상태인지 알려면 DB2GSE.SPATIAL_GEOCODER 카탈로그 뷰(121 페이지의 『DB2GSE.SPATIAL_GEOCODER』에 서술됨)를 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 이 저장 프로시듀어가 등록하는 지오코더가 들어 있는 데이터베이스에 대해 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

입력 매개변수

표 24. db2gse.gse_register_gc 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
gcId	INTEGER	등록할 지오코더의 숫자 식별자. 이 매개변수는 널 입력이 가능하지 않습니다. **주석:** 이 식별자는 데이터베이스 내에서 고유해야 합니다.
gcName	VARCHAR(64)	등록할 지오코더에 대한 간단한 설명. 이 매개변수는 널 입력이 가능하지 않습니다. **주석:** 이 설명은 데이터베이스 내에서 고유 문자열이어야 합니다.
vendorName	VARCHAR(64)	등록할 지오코더를 제공하는 벤더의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
primaryUDF	VARCHAR(256)	등록할 지오코더의 완전한 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
precisionLevel	INTEGER	지오코더가 소스 데이터를 정상적으로 처리하기 위해 소스 데이터가 해당 참조 자료와 일치해야 하는 정도. 이 매개변수는 널 입력이 가능하지 않습니다. **주석:** 정밀도 레벨의 범위는 1 - 100%입니다.

표 24. db2gse.gse_register_gc 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
vendorSpecific	VARCHAR(256)	벤더가 제공하는 기술 정보. 예를 들어, 벤더가 매개변수를 설정하는 데 사용하는 파일의 경로와 이름. 이 매개변수는 널 입력이 가능합니다.
geoArea	VARCHAR(256)	지오코딩될 지리적인 영역. 이 매개변수는 널 입력이 가능합니다.
설명	VARCHAR(256)	벤더가 제공하는 주석. 이 매개변수는 널 입력이 가능합니다.

출력 매개변수

표 25. db2gse.gse_register_gc 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_register_layer

이 저장 프로시듀어를 사용하여 공간 컬럼을 계층으로 등록하십시오. 이 저장 프로시듀어가 처리될 때, 등록될 계층에 대한 정보가 DB2GSE.GEOMETRY_COLUMNS 카탈로그 뷰에 추가됩니다. 120 페이지의 『DB2GSE.GEOMETRY_COLUMNS』에서 이 뷰에 대한 자세한 내용을 참조하십시오.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseRegisterLayer를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 테이블 계층의 경우:
  - 이 계층이 속해 있는 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한.
  - 이 테이블에 대한 CONTROL 또는 ALTER 특권.
- 뷰 계층의 경우:
  - (1) 이 계층에 대해 지오코딩될 주소 데이터 및 (2) 지오코딩으로 인해 생긴 공간 데이터가 들어 있는 기본 테이블(들)에 대한 SELECT 특권.

입력 매개변수

표 26. db2gse.gse_register_layer 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
layerSchema	INTEGER(30)	layerTable 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수는 널 입력이 가능합니다. **주석:** layerSchema 매개변수에 대한 값을 제공하지 않은 경우에는 db2gse.gse_register_layer 저장 프로시듀어가 호출된 사용자 ID를 기본값으로 합니다.

표 26. db2gse.gse_register_layer 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
layerTable	VARCHAR(128)	계층으로 등록될 컬럼이 들어있는 테이블 또는 뷰의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
layerColumn	VARCHAR(128)	계층으로 등록될 컬럼의 이름. 컬럼이 존재하지 않으면 DB2 Spatial Extender가 이를 작성합니다. 이 매개변수는 널 입력이 가능하지 않습니다.
layerTypeName	VARCHAR(64)	계층으로 등록될 컬럼의 데이터 유형. 데이터 유형은 대문자로 지정해야 합니다. 예를 들면 다음과 같습니다. `ST_POINT` 컬럼이 이 저장 프로시듀어가 처리될 때 작성되는 테이블 컬럼인 경우, 이 매개변수는 널 입력이 가능하지 않습니다. 그러나, 컬럼이 테이블이나 뷰의 기존 컬럼인 경우에는 이 매개변수는 널 입력이 가능합니다.
srId	INTEGER	이 계층에 사용된 공간 참조 시스템의 식별자. 이 매개변수는 테이블 계층에 대해 널 입력이 가능하지 않습니다. 뷰 계층을 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다.
geoSchema	VARCHAR(30)	뷰 컬럼을 계층으로 등록할 때 적용됩니다. geoSchema 매개변수는 컬럼이 속해 있는 뷰의 기초가 되는 테이블의 스키마입니다. 뷰 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 테이블 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. **주석:** geoSchema 매개변수에 대한 값을 제공하지 않은 경우에는 layerSchema 매개변수의 값을 기본값으로 합니다.
geoTable	VARCHAR(128)	뷰 컬럼을 계층으로 등록할 때 적용됩니다. geoTable 매개변수는 컬럼이 속해 있는 뷰의 기초가 되는 테이블의 이름입니다. 뷰 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능하지 않습니다. 테이블 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다.

표 26. db2gse.gse_register_layer 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
geoColumn	VARCHAR(128)	뷰 컬럼을 계층으로 등록할 때 적용됩니다. geoColumn 매개변수는 이 뷰 컬럼의 기초가 되는 테이블 컬럼의 이름입니다. 뷰 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능하지 않습니다. 테이블 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다.
nAttributes	SMALLINT	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 컬럼의 번호. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다.
attr1Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 첫 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더를 사용하려는 경우에는 attr1Name 컬럼에 거리(street) 주소를 저장해야 합니다.
attr2Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 두 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더를 사용하려는 경우에는 attr2Name 컬럼에 도시(city)의 이름을 저장해야 합니다.
attr3Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 세 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더를 사용하려는 경우에는 attr3Name 컬럼에 도(state)의 이름 또는 약어를 저장해야 합니다.

표 26. db2gse.gse_register_layer 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
attr4Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 네 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더를 사용하려는 경우에는 attr4Name 컬럼에 우편번호(zip code)를 저장해야 합니다.
attr5Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 다섯 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더는 Attr5Name 컬럼을 무시합니다.
attr6Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 여섯 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더는 Attr6Name 컬럼을 무시합니다.
attr7Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 일곱 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더는 Attr7Name 컬럼을 무시합니다.
attr8Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 여덟 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더는 Attr8Name 컬럼을 무시합니다.

표 26. db2gse.gse_register_layer 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
attr9Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 아홉 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더는 Attr9Name 컬럼을 무시합니다.
attr10Name	VARCHAR(128)	이 계층에 대해 지오코딩될 소스 데이터가 들어 있는 열 번째 컬럼의 이름. 테이블 컬럼을 계층으로 등록하는 경우, 이 매개변수는 널 입력이 가능합니다. 뷰 컬럼을 계층으로 등록할 때는 DB2 Spatial Extender가 이 매개변수를 무시합니다. 기본 지오코더는 Attr10Name 컬럼을 무시합니다.

출력 매개변수

표 27. db2gse.gse_register_layer 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

제한사항

- 뷰 컬럼을 계층으로 등록할 경우에는 이미 계층으로 등록된 테이블 컬럼에 기초해야 합니다.
- 불과 10개의 속성 컬럼에만 등록할 계층에 대해 지오코딩될 데이터가 포함될 수 있습니다.

db2gse.gse_run_gc

이 저장 프로시듀어를 사용하여 지오코더를 일괄처리 모드로 실행하십시오. 50 페이지의 『지오코더를 일괄처리 모드로 실행』에서 이 타스크에 대한 자세한 내용을 참조하십시오.

이 저장 프로시듀어를 호출하기 위한 코드 예를 보려면 샘플 프로그램의 C 함수 gseRunGC를 참조하십시오. 65 페이지의 『제8장 DB2 Spatial Extender용 응용 프로그램 작성』에서 이 프로그램에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 지정된 지오코더가 운영되는 테이블을 포함하는 데이터베이스의 SYSADM 또는 DBADM 권한.
- 이 테이블에 대한 CONTROL 또는 UPDATE 특권.

입력 매개변수

표 28. db2gse.gse_run_gc 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
layerSchema	VARCHAR(30)	layerTable 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수는 널 입력이 가능합니다. **주석:** layerSchema 매개변수에 대한 값을 제공하지 않은 경우에는 db2gse.gse_run_gc 가 호출된 사용자 ID를 기본값으로 합니다.
layerTable	VARCHAR(128)	지오코딩된 데이터가 삽입될 컬럼이 들어 있는 테이블 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
layerColumn	VARCHAR(128)	지오코딩된 데이터가 삽입될 컬럼 이름. 이 매개변수는 널 입력이 가능하지 않습니다.

표 28. db2gse.gse_run_gc 저장 프로시듀어에 대한 입력 매개변수. (계속)

이름	데이터 유형	설명
gcId	INTEGER	실행할 지오코더의 식별자. 이 매개변수는 널 입력이 가능합니다. 등록된 지오코더의 식별자를 알려면 DB2GSE.SPATIAL_GEOCODER 카탈로그 뷰를 참조하십시오.
precisionLevel	INTEGER	지오코더가 소스 데이터를 정상적으로 처리하기 위해 소스 데이터가 해당 참조 자료와 일치해야 하는 정도. 이 매개변수는 널 입력이 가능합니다. **주석:** 정밀도 레벨의 범위는 1 - 100%입니다.
vendorSpecific	VARCHAR(256)	벤더가 제공하는 기술 정보. 예를 들어, 벤더가 매개변수를 설정하는 데 사용하는 파일의 경로와 이름. 이 매개변수는 널 입력이 가능합니다.
whereClause	VARCHAR(256)	WHERE절의 본체. 이는 지오코딩될 레코드 세트에 대한 제한사항을 정의합니다. 절에서는 지오코더가 운용될 테이블에 있는 모든 속성 컬럼을 참조할 수 있습니다. 이 매개변수는 널 입력이 가능합니다.
commitScope	INTEGER	체크점당 레코드 수. 이 매개변수는 널 입력이 가능합니다.

출력 매개변수

표 29. db2gse.gse_run_gc 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_unregist_gc

이 저장 프로시듀어를 사용하여 기본 지오코더가 아닌 다른 지오코더를 등록 해제하십시오.

DB2GSE.SPATIAL_GEOCODER 카탈로그 뷰에서 등록 해제하려는 지오코더에 대한 자세한 내용을 참조하십시오. 121 페이지의 『DB2GSE.SPATIAL_GEOCODER』를 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 등록 해제될 지오코더가 들어 있는 데이터베이스에 대해 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

입력 매개변수

표 30. db2gse.gse_unregist_gc 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
gcId	INTEGER	등록 해제될 지오코더의 식별자. 이 매개변수는 널 입력이 가능하지 않습니다.

출력 매개변수

표 31. db2gse.gse_unregist_gc 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

db2gse.gse_unregist_layer

이 저장 프로시듀어를 사용하여 계층을 등록 해제하십시오. 저장 프로시듀어는 다음과 같이 수행합니다.

- DB2 Spatial Extender 카탈로그 테이블로부터 계층 정의를 제거함.
- DB2 Spatial Extender가 계층의 공간 데이터가 계층의 공간 참조 시스템 요구사항을 따르는지 확인하기 위해 이 계층의 기본 테이블에 위치시킨 점검 제한조건을 지움.
- 주소 데이터가 추가, 변경 또는 제거될 때마다 공간 컬럼을 갱신하는 데 사용되는 트리거를 제거함.

이 저장 프로시듀어가 처리될 때, 계층에 대한 정보가 DB2GSE.GEOMETRY_COLUMNS 메타 뷰에서 제거됩니다. 120 페이지의 『DB2GSE.GEOMETRY_COLUMNS』에서 이 뷰에 대한 자세한 내용을 참조하십시오.

권한

이 저장 프로시듀어를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 테이블 계층의 경우:
  - 이 계층의 기본 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한.
  - 이 테이블에 대한 CONTROL 또는 ALTER 특권.
- 뷰 계층의 경우:
  - (1) 이 계층에 대해 지오코딩될 주소 데이터 및 (2) 지오코딩으로 인해 생긴 공간 데이터가 들어 있는 기본 테이블(들)에 대한 SELECT 특권.

입력 매개변수

표 32. db2gse.gse_unregist_layer 저장 프로시듀어에 대한 입력 매개변수.

이름	데이터 유형	설명
layerSchema	VARCHAR(30)	layerTable 매개변수에 지정된 테이블이 속해 있는 스키마의 이름. 이 매개변수는 널 입력이 가능합니다. **주석:** layerSchema 매개변수에 대한 값을 제공하지 않은 경우에는 db2gse.gse_unregister_layer 저장 프로시듀어가 호출된 사용자 ID를 기본값으로 합니다.
layerTable	VARCHAR(128)	layerColumn 매개변수에 지정된 컬럼이 들어 있는 테이블 이름. 이 매개변수는 널 입력이 가능하지 않습니다.
layerColumn	VARCHAR(128)	등록 해제할 계층으로 정의되어 있는 공간 컬럼의 이름. 이 매개변수는 널 입력이 가능하지 않습니다.

출력 매개변수

표 33. db2gse.gse_unregist_layer 저장 프로시듀어에 대한 출력 매개변수.

이름	데이터 유형	설명
msgCode	INTEGER	이 저장 프로시듀어의 호출자가 리턴할 수 있는 메시지와 연관된 코드.
예약됨	VARCHAR(1024)	DB2 Spatial Extender 서버에서 구성될 때의 완전한 오류 메시지

제한사항

뷰 계층으로 정의된 뷰 컬럼이 테이블 계층으로 정의된 테이블 컬럼에 기초한 경우에는, 뷰 계층을 등록 해제할 때까지 이 테이블 계층을 등록 해제할 수 없습니다.

제10장 메세지

이 장에서는 DB2 Spatial Extender가 사용자에게 리턴하는 메시지에 대해 설명합니다. 각 메시지들은 식별자를 가지고 있습니다. 영문자 E로 끝나는 식별자는 오류 메시지이고, W로 끝나는 식별자는 경고 메시지, I로 끝나는 식별자는 일반 정보입니다.

DBA7200E 지오코더에 대한 입력으로 11개 이상의 컬럼이 선택되었습니다.

설명: 지오코더에 대한 입력으로 최대 10개의 컬럼을 선택할 수 있습니다.

사용자 응답: 선택한 컬럼 상자에서 10개 이하의 이름을 나열할 때까지 선택한 컬럼 상자에서 사용 가능한 컬럼 상자로 컬럼 이름을 이동하십시오.

DBA7201E 데이타베이스가 Spatial Extender 조작에 대해 사용 불가능하게 됩니다.

설명: 사용자가 Spatial Extender를 사용하기 전에 Spatial Extender에서 데이터베이스를 사용할 수 있어야 합니다.

사용자 응답: 데이터베이스를 마우스 오른쪽 버튼으로 클릭하고 메뉴에서 Spatial Extender —> 사용을 선택하십시오.

GSE0000I 조작이 정상적으로 완료되었습니다.

GSE0001E Spatial Extender가 사용자 ID "<user-id>" 하에서 요청된 조작 ("<operation-name>")을 수행할 수 없습니다.

설명: 조작을 수행하기 위한 특권이나 권한을 보유하고 있지 않는 사용자 ID 하에서 이 조작을 요청했습니다.

사용자 응답: 문서에서 올바른 권한 부여가 무엇인지 찾아 보거나 Spatial Extender 관리자에게서 이를 받으십시오.

GSE0002E "<value>"가 "<argument-name>" 인수에 유효한 값이 아닙니다.

설명: 사용자가 입력한 값이 틀렸거나 철자가 맞지 않습니다.

사용자 응답: 지정해야 할 값 또는 값의 범위를 알려면 문서를 찾아 보거나 Spatial Extender 관리자에게 문의하십시오.

GSE0003E Spatial Extender가 인수 "<argument-name>"이 지정되지 않아 요청된 조작을 수행할 수 없습니다.

설명: 이 조작에 필요한 인수를 지정하지 않았습니다.

사용자 응답: 원하는 값으로 인수 "<argument-name>"을 지정하십시오. 그런 다음, 조작을 다시 요청하십시오.

GSE0004W 인수 "<argument-name>"이 평가되지 않았습니다.

설명: 사용자가 요청한 조작에서 인수 "<argument-name>"을 사용할 수 없습니다.

사용자 응답: 아무 것도 필요하지 않습니다.

GSE0005E Spatial Extender가 "<object-name>"이라는 오브젝트를 작성하라는 사용자의 요청을 처리할 수 없습니다.

설명: 오브젝트 "<object-name>"이 이미 존재하거나 오브젝트를 작성할 수 있는 적절한 사용권한을 가지고 있지 않습니다. 테이블, 컬럼, 트리거, 색인, 파일 또는 다른 종류의 오브젝트들이 오브젝트가 될 수 있습니다.

사용자 응답: "<object-name>"이 사용자가 원하는 오브젝트이면 아무 것도 수행하지 마십시오. 그렇지 않은 경우에는 이름을 올바로 지정하고 오브젝트를 작성할 수 있는 올바른 사용권한을 가지고 있는지 검증하십시오.

GSE0006E Spatial Extender가 "<object-name>"이라는 사용 가능하거나 등록된 오브젝트에 대해 요청된 조작을 수행할 수 없습니다.

설명: 오브젝트 "<object-name>"이 이미 사용 가능하거나 등록된 상태일 수도 있고 이미 존재할 수도 있습니다. 계층, 색인, 공간 참조 시스템, 좌표 시스템, 지오코더 또는 다른 종류의 오브젝트들이 오브젝트가 될 수 있습니다.

사용자 응답: 오브젝트 "<object-name>"이 존재하는지 확인하고 요청을 다시 제출하십시오.

GSE0007E 아직 사용 가능하지 않거나 등록되지 않은 "<object-name>", 오브젝트에 대해 Spatial Extender가 요청된 조작을 수행할 수 없습니다.

설명: 오브젝트 "<object-name>"이 사용 가능하지 않거나 등록되지 않았습니다. 계층, 색인, 공간 참조 시스템, 공간 좌표 시스템, 지오코더 또는 다른 종류의 오브젝트들이 오브젝트가 될 수 있습니다.

사용자 응답: 오브젝트 "<object-name>"을 사용 가능하게 하거나 등록하십시오. 그런 다음, 요청을 다시 제출하십시오.

GSE0008E 예기치 못한 SQL 오류 ("<sql-error-message>")가 발생했습니다.

사용자 응답: SQL 오류 메시지 "<sql-error-message>"의 SQLCODE와 관련된 세부 메시지를 찾아 보십시오. 필요하다면 IBM 서비스 담당자에게 문의하십시오.

GSE0009E 이미 존재하는 "<object-name>"이라는 오브젝트에 대해 요청된 조작을 수행할 수 없습니다.

설명: "<object-name>"이 운영 체제 또는 데이터베이스에 이미 존재합니다. 파일, 테이블, 뷰, 컬럼, 색인, 트리거 또는 다른 종류의 오브젝트들이 오브젝트가 될 수 있습니다.

사용자 응답: 오브젝트에 액세스하려고 시도할 때 오브젝트를 올바로 지정했는지 확인하십시오. 필요하다면 오브젝트를 삭제하십시오.

GSE0010E 존재하지 않는 "<object-name>"이라는 오브젝트에 대해 요청된 조작을 수행할 수 없습니다.

설명: "<object-name>"이 데이터베이스 또는 운영 체제에 존재하지 않습니다. 파일, 테이블, 뷰, 컬럼,

색인, 트리거 또는 다른 종류의 오브젝트들이 오브젝트가 될 수 있습니다.

사용자 응답: 오브젝트에 액세스할 수 있는 올바른 사용권한을 가지고 있는지 확인하십시오. 해당 사용권한을 가지고 있고 오브젝트가 존재하지 않는다면, 오브젝트를 작성해야 합니다.

GSE0011E Spatial Extender가 오브젝트 "<object-name>"을 사용할 수 없거나 등록할 수 없습니다.

설명: "<object-name>"이 다른 오브젝트에 종속적입니다. "<object-name>"은 공간 참조 시스템, 계층, 지오코더 또는 다른 종류의 오브젝트들이 될 수 있습니다.

사용자 응답: 문서를 참조하여 어떤 종류의 오브젝트들 "<object-name>"이 종속될 수 있는지 찾아 보십시오. 그런 다음, "<object-name>"이 종속적인 특정 오브젝트를 제거하십시오.

GSE0012E 완전한 공간 컬럼 "<layer-schema.layer-name.layer-column>"이 테이블 계층으로 등록되지 않아 Spatial Extender가 요청을 처리할 수 없습니다.

설명: 완전한 공간 컬럼 "<layer-schema.layer-name.layer-column>"은 사용자가 이와 관련된 특정 조작(예: 색인을 사용하거나 지오코더를 사용하여 일괄처리 모드로 이를 데이터 상주시키거나 자동으로 갱신함)을 수행하기 전에 테이블 계층으로 등록되어야 합니다.

사용자 응답: Spatial Extender 카탈로그의 DB2GSE.GEOMETRY_COLUMNS를 점검하여 완전한 공간 컬럼 "<layer-schema.layer-name.layer-column>"이 테이블 계층으로 등록된 상태인지 확인하십시오. 또한 이 컬럼이 포함된 테이블에 유효한 해당 속성 컬럼도 포함되어 있는지 확인하십시오.

GSE0013E 데이타베이스가 공간 데이터 조작에 대해 사용 불가능하게 됩니다.

설명: 데이타베이스가 공간 데이터 조작에 대해 사용 불가능하게 됩니다. 그러므로 Spatial Extender 카탈로그가 존재하지 않습니다.

사용자 응답: 공간 데이터 조작용으로 데이터베이스를 사용할 수 있도록 하십시오.

GSE0014E 데이터베이스가 공간 데이터 조작용으로 이미 사용 가능합니다.

설명: 데이터베이스가 공간 데이터 조작용으로 이미 사용 가능합니다.

사용자 응답: 예상했던 대로 데이터베이스가 사용 가능한지 확인하십시오. 필요하다면 데이터베이스를 사용하지 마십시오.

GSE0498E 다음과 같은 오류가 발생했습니다: "<error-message>".

GSE0499W Spatial Extender가 다음과 같은 경고를 발행했습니다: "<warning-message>".

GSE0500E 지정된 조작 모드 ("<operation-mode>")가 올바르지 않습니다.

설명: 사용자가 요청한 조작에서 지정된 모드를 지원하지 않습니다.

사용자 응답: 조작에서 어떤 모드를 지원하는지 알려면 문서를 찾아 보십시오.

GSE1001E Spatial Extender가 이름이 "<schema-name.view-name.column-name>"이면서 공간 컬럼 "<schema-name.table-name.column-name>"에 기초한 뷰 계층을 등록할 수 없습니다.

설명: 지정된 공간 컬럼("<schema-name.table-name.column-name>")이 테이블 계층으로 등록되지 않았습니다.

사용자 응답: 컬럼 "<schema-name.table-name.column-name>"을 테이블 계층으로 등록하십시오.

GSE1002E Spatial Extender가 이름이 "<schema-name.view-name.column-name>"이면서 테이블 "<schema-name.table-name>"에 기초한 뷰 계층을 등록할 수 없습니다.

설명: 지정된 테이블("<schema-name.table-name>")이 직접 또는 간접적으로 뷰 "<schema-name.view-name.column-name>"의 기초가 되지 않았습니다.

사용자 응답: 뷰 "<schema-name.view-name.column-name>"에 대한 기본 테이블이 무엇인지 찾아서 이 테이블을 지정하십시오.

GSE1003E Spatial Extender가 "<schema-name.object-name>"이라는 테이블 또는 뷰의 "<column-name>"이라는 컬럼에 액세스할 수 없습니다.

설명: 테이블 또는 뷰 "<schema-name.object-name>"에 "<column-name>"이라는 컬럼이 없습니다.

사용자 응답: 테이블 또는 뷰 "<schema-name.object-name>"의 정의를 점검하여 원하는 컬럼의 올바른 이름을 알아 보십시오.

GSE1004E Spatial Extender가 완전한 공간 컬럼 "<schema-name.table-name.column-name>"을 테이블 계층으로 등록할 수 없습니다.

설명: 컬럼 "<schema-name.table-name.column-name>"에 공간 데이터 유형이 없거나 기본 테이블과 연관되지 않았습니다.

사용자 응답: 컬럼 "<schema-name.table-name.column-name>"에 대한 공간 데이터 유형을 정의하거나 이 컬럼이 지역 기본 테이블의 일부분인지 확인하십시오.

GSE1005E 뷰 계층에 대해 지정된 공간 참조 시스템("<view-layer-spatial-reference-id>")이 이 계층의 기초가 되는 테이블 계층에 대해 사용된 공간 참조 시스템 ("<table-layer-spatial-reference-id>")과 서로 다릅니다.

설명: 뷰 계층의 공간 참조 시스템은 기초가 되는 테이블 계층의 공간 참조 시스템과 동일해야 합니다.

사용자 응답: 뷰 계층에 대해 기초가 되는 테이블 계층의 공간 참조 시스템을 지정하십시오.

GSE1006E "<spatial-reference-id>"가 올바르지 않은 공간 참조 시스템 ID이기 때문에, Spatial Extender가 사용자가 요청한 계층을 등록할 수 없습니다.

설명: 지정된 공간 참조 시스템("<spatial-reference-id>")이 사용되지 않거나 등록되지 않았습니다.

사용자 응답: 공간 참조 시스템을 사용 가능하게 하거나 등록하십시오. 그런 다음, 요청을 다시 제출하여 계층을 등록하십시오.

GSE1007E Spatial Extender가 공간 컬럼 (“<column-name>”)을 테이블 “<schema-name.table-name>”에 비정상적으로 추가하려고 시도할 때 SQL 오류(SQLSTATE “<sqlstate>”)가 발생했습니다.

사용자 응답: SQLSTATE “<sqlstate>”와 연관된 메시지를 찾아 보십시오.

GSE1008E 뷰 계층의 공간 데이터 유형 “<layer-column-type>”이 기초가 되는 테이블 계층 “<geo-schema.geo-name.geo-column>”의 공간 데이터 유형 “<geo-column-type>”과 일치하지 않기 때문에 DB2 Spatial Extender가 뷰 계층 “<layer-schema.layer-name.layer-column>”을 등록할 수 없습니다.

설명: 뷰 계층 “<layer-schema.layer-name.layer-column>”의 공간 데이터 유형은 계층의 기초가 되는 테이블 계층 “<geo-schema.geo-name.geo-column>”의 공간 데이터 유형과 일치해야 합니다. 이 두 가지 데이터 유형간의 불일치로 인해 공간 데이터가 처리될 때 혼란이 일어날 수 있습니다.

사용자 응답: 뷰 계층의 공간 데이터 유형과 기초가 되는 테이블 계층이 동일한지 확인하십시오.

GSE1020E “<’spatial-reference-id>”는 올바르지 않은 공간 참조 시스템 ID입니다.

설명: 식별자가 “<’spatial-reference-id>”인 공간 참조 시스템을 사용할 수 없습니다.

사용자 응답: 지정된 공간 참조가 사용되었는지 확인하십시오.

GSE1021E Spatial Extender가 공간 참조 시스템 “<spatial-reference-id>”를 사용할 수 없는데 이는 해당 공간 좌표 시스템 ID “<spatial-coordinate-id>”가 올바르지 않기 때문입니다.

설명: 식별자가 “<spatial-coordinate-id>”인 좌표 시스템이 Spatial Extender 카탈로그에 정의되지 않습니다.

사용자 응답: Spatial Extender 카탈로그의 DB2GSE.COORD_REF_SYS 뷰를 점검하여 좌표 시스템 식별자 “<spatial-coordinate-id>”를 검증하십시오.

GSE1030E “<schema-name.table-name>”이 기본 테이블이 아니기 때문에 Spatial Extender가 그 용도로 지오코더를 사용할 수 없습니다.

설명: 지오코딩하려는 소스 데이터가 들어 있는 오브젝트가 기본 테이블이어야 합니다.

사용자 응답: 지오코딩하려는 소스 데이터가 들어 있는 컬럼이 기본 테이블의 일부분인지 확인하십시오.

GSE1031E Spatial Extender가 계층 “<layer-schema.layer-name.layer-column>”에 대해 자동으로 작성 모드로 운용되도록 지오코더 “<geocoder-id>”를 작동시킬 수 없습니다.

설명: 가능한 설명은 다음과 같습니다.

- 지오코더가 이미 계층 “<layer-schema.layer-name.layer-column>”을 자동으로 갱신할 수 있습니다.
- 지오코더가 이 계층에 대해 임시로 유효성 검사를 하지 않았습니다.

- 소스 데이터에 대한 어떤 컬럼도 이 계층에 대해 정의되지 않았습니다.

사용자 응답: 지오코더가 임시로 유효성 검사를 하지 않은 경우에는 지오코더를 작동시켜 자동으로 ″재작성″ 모드로 운용되게 하십시오.

GSE1032E Spatial Extender가 계층 “<layer-schema.layer-name.layer-column>”에 대해 자동으로 재작성 모드로 운용되도록 지오코더 “<geocoder-id>”를 작동시킬 수 없습니다.

설명: 가능한 설명은 다음과 같습니다.

- 지오코더가 이미 계층 “<layer-schema.layer-name.layer-column>”을 자동으로 갱신할 수 있습니다.
- 지오코더가 예전에 이 계층에 대해 유효성 검사를 하지 않았습니다.
- 소스 데이터에 대한 어떤 컬럼도 이 계층에 대해 정의되지 않았습니다.

사용자 응답: 지오코더가 예전에 제거 모드로 사용될 수 없었거나 이 계층에 대해 정의된 적이 없는 경우에는, 자동으로 ″작성″ 모드로 운용되도록 지오코더를 작동시키십시오.

GSE1033E Spatial Extender가 계층 “<layer-schema.layer-name.layer-column>”에 대한 컬럼이 들어 있는 테이블에 트리거를 추가하려고 시도할 때 SQL 오류가 발생했습니다(SQLSTATE “<sqlstate>”).

설명: 트리거의 목적은 지오코더의 입력을 가져오게 될 속성 컬럼과 그 출력이 놓이게 될 공간 컬럼 사이에 데이터 무결성을 유지보수하는 것입니다. DB2가 이 트리거를 작성하려고 시도할 때 SQL 오류가 발생했습니다.

사용자 응답: SQLSTATE “<sqlstate>”와 연관된 메시지를 찾아 보십시오.

GSE1034E Spatial Extender가 계층 “<layer-schema.layer-name.layer-column>”에 대해 제거 모드로 지오코더 “<geocoder-id>”를 사용할 수 없습니다.

설명: 가능한 설명은 다음과 같습니다.

- 지오코더가 계층 “<layer-schema.layer-name.layer-column>”을 자동으로 갱신할 수 없습니다.
- 지오코더를 제거 모드로 사용할 수 없습니다.

사용자 응답: 지오코더를 사용 불가능하게 하기 전에 지오코더의 상태를 판별하십시오. 예를 들어, 지오코더가 등록된 상태입니까? 사용 가능한 상태입니까? 그런 다음, 제거 모드로 사용할 수 없게 해야 하는지 여부를 판별하십시오. 예를 들어, 사용 가능하지 않은 경우에는 이를 사용 불가능 상태로 만들 필요가 전혀 없습니다.

GSE1035E Spatial Extender가 계층 "<layer-schema.layer-name. layer-column>"에 대해 비유효성 확인 모드로 지오코더 "<geocoder-id>"를 사용 불가능하게 할 수 없습니다.

설명: 가능한 설명은 다음과 같습니다.

- 지오코더가 계층 "<layer-schema.layer-name. layer-column>"을 자동으로 갱신할 수 없습니다.
- 지오코더를 비유효성 확인 모드 또는 제거 모드로 사용하지 않았습니다.

사용자 응답: 지오코더를 사용 불가능하게 하기 전에 지오코더의 상태를 판별하십시오. 예를 들어, 지오코더가 등록된 상태입니까? 사용 가능한 상태입니까? 그런 다음, 비유효성 모드로 사용할 수 없게 해야 하는지 여부를 판별하십시오. 예를 들어, 이미 비유효성 모드로 사용 불가능한 경우에는 지오코더를 또 한번 이 모드로 사용 불가능하게 만들 필요가 없습니다.

GSE1036E Spatial Extender가 계층 "<layer-schema.layer-name. layer-column>"에 대한 컬럼이 들어 있는 테이블에서 트리거를 제거하려고 시도할 때 SQL 오류가 발생했습니다(SQLSTATE "<sqlstate>").

설명: 트리거는 지오코더의 입력을 가져오게 될 속성 컬럼과 그 출력이 놓이게 될 공간 컬럼 사이에 데이터 무결성을 유지보수하기 위해 작성되었습니다. DB2가 이 트리거를 제거하려고 시도할 때 SQL 오류가 발생했습니다.

사용자 응답: SQLSTATE "<sqlstate>"와 연관된 메시지를 찾아 보십시오.

GSE1037E 얼마나 많은 속성 컬럼들이 이 계층에 대해 소스 데이터를 제공할 것인지 지정하는 인수에 틀린 값 "<number-of-attributes>"가 지정되었기 때문에, Spatial Extender가 테이블 계층 "<layer-schema.layer-name. layer-column>"에 대해 소스 데이터를 지오코딩할 수 없습니다.

설명: 이 계층과 연관된 속성 컬럼의 수가 틀리게 지정되었거나 하나 이상의 이 컬럼 이름이 틀리게 지정되었습니다.

사용자 응답: 이 계층이 관련 속성 컬럼의 올바른 수와 이름으로 등록되었는지 확인하거나 지오코더에 대한 입력 및 출력 데이터의 정당성을 검증하십시오.

GSE1038E Spatial Extender가 테이블 계층 "<layer-schema.layer-name. layer-column>"에 대한 소스 데이터를 일괄처리 모드로 지오코딩하려고 시도할 때 SQL 오류가 발생했습니다(SQLSTATE "<sqlstate>").

사용자 응답:

- SQLSTATE "<sqlstate>"와 연관된 메시지를 찾아 보십시오.
- 이 계층의 내용과 기본 UDF 인수가 올바로 정의되었는지 확인하십시오.

GSE1050E 지정된 격자 *크기*(“<grid-size>”)가 첫번째 격자 레벨에 맞지 않습니다.

설명: 첫번째 격자 레벨에 대한 격자 *크기*로 0 또는 음수를 지정했습니다.

사용자 응답: 격자 *크기*로 양수를 지정하십시오.

GSE1051E 지정된 격자 *크기*(“<grid-size>”)가 두 번째 및 세 번째 격자 레벨에 맞지 않습니다.

설명: 두 번째 또는 세 번째 격자 레벨에 대한 격자 *크기*로 음수를 지정했습니다.

사용자 응답: 격자 *크기*로 0 또는 양수를 지정하십시오.

GSE1052E Spatial Extender가 테이블 계층 “<layer-schema.layer-name. layer-column>”에 대한 공간 색인 “<index-schema.index-column>”을 작성하려고 시도할 때 SQL 오류가 발생했습니다(SQLSTATE “<sqlstate>”).

사용자 응답:

- 공간 색인이 올바로 지정되고 공간 컬럼에 관련된 색인이 없는지 확인하십시오.
- SQLSTATE “<sqlstate>”와 연관된 메시지를 찾아 보십시오.

GSE1500I 소스 레코드 “<record-number>”가 정상적으로 지오코딩되었습니다.

설명: 속성 데이터가 들어 있는 레코드가 정상적으로 지오코딩되었습니다.

GSE1501W 소스 레코드 “<record-number>”가 지오코딩되지 않았습니다.

설명: 정밀도 레벨이 너무 높습니다.

사용자 응답: 보다 낮은 정밀도 레벨로 지오코딩하십시오.

GSE1502W 소스 레코드 “<record-number>”를 찾을 수 없습니다.

사용자 응답: 레코드가 데이터베이스에 존재하는지 여부를 판별하십시오.

GSE2001E 지정된 전송 파일(“<filename>”)이 유효하지 않습니다.

사용자 응답: 지정된 파일이 SDE 전송 파일이고 경로 이름이 올바로 지정되었는지 검증하십시오.

GSE2002E 제공된 SQL WHERE절(“<SQL-where-clause>”)이 유효하지 않습니다.

사용자 응답: WHERE절의 SQL 구문이 올바른지 철자 오류 및 올바르지 않은 컬럼 이름이 없는지 점검하십시오.

GSE2003E 제공된 형상값이 적법하지 않습니다.

사용자 응답: 제공된 형상이 공간 컬럼의 지정된 유형과 일치하는지 점검하십시오. 유형이 일치하거나 호환가능하면 기하학 양식이 잘못된 것입니다. 다각형, 단일 점 원호(arc) 등이 겹치는지 점검하십시오.

GSE2004E 전송 파일 스키마가 지정된 계층의 스키마와 호환되지 않습니다.

사용자 응답: 스키마와 계층 이름이 제대로 지정되었는지 점검하십시오. 스키마들이 일치하지 않으면, 데이터를 새로운 테이블로서 로드하고 스키마간의 차이점을 해결하십시오.

GSE2005E 전송 파일 기하학 유형이 지정된 계층의 기하학 유형과 호환되지 않습니다.

사용자 응답: 스키마와 계층 이름이 제대로 지정되었는지 점검하십시오.

GSE2006E "<filename>"이라는 파일에 대해 입출력 오류가 발생했습니다.

사용자 응답: 파일이 존재하는지, 파일에 대해 해당 액세스권을 가지고 있는지, 다른 사용자가 파일을 사용하고 있지 않는지 검증하십시오.

GSE2007E 속성 변환 오류가 발생했습니다.

사용자 응답: 테이블 내의 모든 속성 유형들이 지원되는지 점검하십시오. 예를 들어, BLOB 데이터는 형상 파일에서 지원되지 않습니다. 범위를 벗어난 데이터 값 또는 날짜가 틀린 잘못된 데이터 값도 점검하십시오.

GSE2008E 가져오기/내보내기 기능을 수행하는 데 메모리가 부족합니다.

사용자 응답: 사용 가능한 메모리가 충분한지 검증하십시오.

제11장 카탈로그 뷰

DB2 Spatial Extender의 카탈로그 뷰에는 다음에 대한 메타데이터가 들어 있습니다.

- 사용할 수 있는 좌표 시스템. 『DB2GSE.COORD_REF_SYS』에서 이 시스템의 식별자 및 주석 텍스트와 같은 내용을 참조하십시오.
- 계층으로 등록된 공간 컬럼. 120 페이지의 『DB2GSE.GEOMETRY_COLUMNS』에서 이 컬럼의 이름, 데이터 유형 및 관련 공간 참조 시스템과 같은 내용을 참조하십시오.
- 사용할 수 있는 지오코더. 121 페이지의 『DB2GSE.SPATIAL_GEOCODER』에서 이 지오코더의 식별자와 지오코더가 처리하는 위치가 들어 있는 영역과 같은 내용을 참조하십시오.
- 사용할 수 있는 공간 참조 시스템. 121 페이지의 『DB2GSE.SPATIAL_REF_SYS』에서 공간 참조 시스템의 식별자 및 설명과 같은 내용을 참조하십시오.

DB2GSE.COORD_REF_SYS

공간 데이터 조작에 대해 데이터베이스를 사용할 때, DB2 Spatial Extender는 사용자가 카탈로그 테이블에 사용할 수 있는 좌표 시스템을 등록합니다. 이 테이블에서 선택한 컬럼들은 표34에 서술된 DB2GSE.COORD_REF_SYS 카탈로그 뷰를 비교합니다.

표 34. DB2GSE.COORD_REF_SYS 카탈로그 뷰의 컬럼들

이름	데이터 유형	널 입력 가능?	Content
SCID	INTEGER	아니오	이 좌표 시스템에 대한 고유 숫자 식별자.
SC_NAME	VARCHAR(64)	아니오	이 좌표 시스템의 이름.
AUTH_NAME	VARCHAR(256)	예	컴파일된 이 좌표 시스템이 지지하는 소속의 이름으로 예를 들어, (EPSG)European Petroleum Survey Group 등이 있습니다.

표 34. DB2GSE.COORD_REF_SYS 카탈로그 뷰의 컬럼들 (계속)

이름	데이터 유형	널 입력 가능?	Content
AUTH_SRID	INTEGER	예	AUTH_NAME 컬럼에 지정된 소속에 의해 이 좌표 시스템에 지정된 숫자 식별자.
DESC	VARCHAR(256)	예	이 좌표 시스템의 설명.
SRTEXT	VARCHAR(2048)	아니오	이 좌표 시스템에 대한 주석 텍스트.

DB2GSE.GEOMETRY_COLUMNS

계층을 작성할 때, DB2 Spatial Extender는 카탈로그 테이블에 계층의 식별자와 그에 관한 정보를 기록하여 계층을 등록합니다. 이 테이블에서 선택한 컬럼들은 표 35에 서술된 DB2GSE.GEOMETRY_COLUMNS 카탈로그 뷰를 비교합니다.

표 35. DB2GSE.GEOMETRY_COLUMNS 카탈로그 뷰의 컬럼들

이름	데이터 유형	널 입력 가능?	Content
LAYER_CATALOG	VARCHAR(30)	예	이 계층의 완전한 이름.
LAYER_SCHEMA	VARCHAR(30)	아니오	이 계층으로 등록된 컬럼이 들어 있는 테이블 또는 뷰의 스키마.
LAYER_NAME	VARCHAR(128)	아니오	이 계층으로 등록된 컬럼이 들어 있는 테이블 또는 뷰의 이름.
LAYER_COLUMN	VARCHAR(30)	아니오	이 계층으로 등록된 컬럼의 이름.
GEOMETRY_TYPE	INTEGER	아니오	이 계층으로 등록된 컬럼의 데이터 유형.
SRID	INTEGER	아니오	이 계층으로 등록된 컬럼 내의 값에 사용된 공간 참조 시스템의 식별자.
STORAGE_TYPE	INTEGER	예	DB2가 이 계층으로 등록된 컬럼 내의 값을 저장하는 방법에 관한 정보. 예를 들어, STORAGE_TYPE에 있는 데이터는 값이 대형 오브젝트(LOB)로 또는 abstract 데이터 유형(ADT)의 인스턴스로 저장됨을 나타냅니다.

DB2GSE.SPATIAL_GEOCODER

사용 가능한 지오코더들은 카탈로그 테이블에 등록됩니다. 이 테이블에서 선택한 컬럼들은 표36에 서술된 DB2GSE.SPATIAL_GEOCODER 카탈로그 뷰를 비교합니다.

표 36. DB2GSE.SPATIAL_GEOCODER 카탈로그 뷰의 컬럼들

이름	데이터 유형	널 입력 가능?	내용
GCID	INTEGER	아니오	이 지오코더의 숫자 식별자.
GC_NAME	VARCHAR(64)	아니오	이 지오코더에 대한 간단한 설명.
VENDOR_NAME	VARCHAR(128)	아니오	이 지오코더를 제공한 벤더의 이름.
PRIMARY_UDF	VARCHAR(256)	아니오	이 지오코더의 완전한 이름.
PRECISION_LEVEL	INTEGER	아니오	지오코더가 소스 데이터를 정상적으로 처리하기 위해 소스 데이터가 해당 참조 자료와 일치해야 하는 정도.
VENDOR_SPECIFIC	VARCHAR(256)	예	벤더가 이 지오코더가 지원하는 특수 매개변수를 설정하는 데 사용할 수 있는 파일의 경로 및 이름.
GEO_AREA	VARCHAR(256)	예	지오코딩될 위치가 들어 있는 지리적 영역.
DESCRIPTION	VARCHAR(256)	예	벤더가 제공하는 주석.

DB2GSE.SPATIAL_REF_SYS

공간 참조 시스템을 작성할 때, DB2 Spatial Extender는 카탈로그 테이블에 공간 참조 시스템의 식별자와 그에 관한 정보를 기록하여 그를 등록합니다. 이 테이블에서 선택한 컬럼들은 표37에 서술된 DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰를 비교합니다.

표 37. DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰의 컬럼들

이름	데이터 유형	널 입력 가능?	Content
SRID	INTEGER	아니오	이 공간 참조 시스템에 대한 사용자 정의 식별자.
SR_NAME	VARCHAR(64)	아니오	이 공간 참조 시스템의 이름.
SCID	INTEGER	아니오	이 공간 참조 시스템의 기초가 되는 좌표 시스템에 대한 숫자 식별자.

표 37. DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰의 컬럼들 (계속)

이름	데이터 유형	널 입력 가능?	Content
SC_NAME	VARCHAR(64)	아니오	이 공간 참조 시스템의 기초가 되는 좌표 시스템의 이름.
AUTH_NAME	VARCHAR(256)	예	이 공간 참조 시스템에 대한 표준을 설정하는 소속의 이름.
AUTH_SRID	INTEGER	예	AUTH_NAME 컬럼에 지정된 소속에서 이 공간 참조 시스템에 할당하는 식별자.
SRTEXT	VARCHAR(2048)	아니오	이 공간 참조 시스템에 대한 주석 텍스트.

제12장 공간 색인

공간 컬럼에는 2차원 지리적 데이터가 들어 있기 때문에, 그 컬럼들을 조회하는 응용프로그램에는 주어진 extent 내에 있는 모든 기하학을 빠르게 식별하는 색인 기법이 필요합니다. 이러한 이유로 DB2 Spatial Extender는 격자를 기본으로 한 3열 공간 색인을 제공합니다.

이 장에서는 이러한 색인 종류에 대해 설명하고 그 사용에 대한 지침도 제공합니다. 다루게 되는 주제는 다음과 같습니다.

샘플 프로그램 조각

색인이 작성되는 방법과 SQL에 어떻게 사용되는지에 대한 다음의 예를 고려해 봅시다. *SQL 참조서*에서 CREATE INDEX 및 CREATE INDEX EXTENSION 명령에 대한 자세한 내용을 참조하십시오. 색인이 작성된 후에는 공간 함수와 관련 술어를 사용하는 표준 DDL 및 DML 명령문을 발행할 수 있습니다.

```
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 트리 색인

공간 색인화 기법은 전형적인 계층 B 트리 색인에 기초하지만 상당 부분이 다릅니다. 공간 색인은 2차원 공간 컬럼을 색인화하도록 설계된 *격자 색인화*를 이용합니다. B 트리 색인은 1차원 데이터만을 핸들할 수 있고 GIS 정보와는 함께 사용될 수 없습니다. 이 절에서는 B 트리 색인이 구조화되고 사용되는 방법에 대해 설명합니다.

루트 노드라고 하는 B 트리 색인의 맨 위 레벨에는 다음 레벨에 있는 각 노드에 대한 키가 하나씩 들어 있습니다. 각 키의 값은 다음 레벨에서의 해당 노드에 대한 기존의 키값 중 가장 큰 키값입니다. 기본 테이블에 있는 값의 수에 따라 여러 중간 노드가 필요할 수도 있습니다. 이 노드는 루트 노드와 실제 기본 테이블 행 ID를 보유하고 있는 리프 노드 사이에 브릿지를 형성합니다.

데이터베이스 관리 프로그램은 루트 노드에서 시작하는 B 트리 색인을 검색합니다. 이는 기본 테이블의 행 ID가 있는 리프 노드에 도달할 때까지 중간 노드들을 거쳐 계속됩니다.

공간 컬럼의 2차원 특성에는 공간 색인의 구조가 필요하기 때문에 B 트리 색인이 공간 컬럼에 적용될 수 없습니다. 동일한 이유로 사용자는 공간 색인을 비공간 컬럼에 적용할 수 없습니다. 더구나, 공간 색인은 모든 종류의 복합 컬럼에 적용될 수 없습니다.

공간 색인 작성 방법

공간 색인을 작성하는 방법에는 여러 가지가 있습니다.

- 공간 색인 작성 창에서 작성하는 방법. 59 페이지의 『제6장 공간 색인 작성』에서 자세한 내용을 참조하십시오.
- 응용프로그램에서 db2gse.gse_enable_idx 저장 프로시듀어를 호출하는 방법. 75 페이지의 『제9장 저장 프로시듀어』에서 이 저장 프로시듀어에 대한 자세한 내용을 참조하십시오.

- USING절에 **db2 create index** 명령과 함께 **spatial_index** 함수를 발행하십시오. 예를 들면 다음과 같습니다.

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

공간 데이터를 사용하려면 데이터베이스 설계자가 공간 데이터의 상대적인 *크기* 분포를 알고 있어야 합니다. 설계자는 공간 색인을 작성할 최적의 *크기*와 격자 레벨 수를 결정해야 합니다.

셀 *크기*를 증가시켜 격자 레벨 <격자 레벨 1>, <격자 레벨 2> 및 <격자 레벨 3>을 입력합니다. 이리하여 두 번째 레벨이 첫번째 셀 *크기* 보다 커야 하고 세 번째는 두 번째 보다 커야 합니다. 첫번째 격자 레벨은 필수이지만 배정밀도 0 값(0.0e0)을 갖는 두 번째 및 세 번째는 사용하지 않을 수 있습니다.

공간 색인의 생성 방법

공간 색인은 *외피*를 사용하여 생성됩니다. 외피는 기하학 그 자체이고 기하학의 최소 및 최대 X와 Y extent를 나타냅니다. 대부분의 기하학 경우에 외피는 상자이지만 수평 및 수직 선스트링의 경우에 외피는 2점 선스트링입니다. 점의 경우에는 외피가 점 그 자체입니다. 139 페이지의 『외피』에서 외피에 대한 자세한 내용을 참조하십시오.

공간 색인은 각 기하학의 외피와 격자의 교차점에 대해 하나 이상의 항목을 만들므로써 공간 컬럼에 대해 구축됩니다. 교차는 기하학의 내부 ID와 교차된 격자 셀의 최소 X 및 Y 좌표로서 기록됩니다. 예를 들어, 126 페이지의 그림7에서는 좌표 (20,30), (30,30), (40,30), (20,40), (30,40), (40,40), (20,50), (30,50) 및 (40,50)에 있는 격자를 교차합니다. 126 페이지의 표38에서 126 페이지의 그림7에 있는 모든 기하학에 대한 최소 X 및 Y 좌표에 대한 자세한 내용을 참조하십시오.

다중 격자 레벨이 존재하는 경우에는, DB2 Spatial Extender가 가능한 가장 낮은 격자 레벨을 사용하려고 시도합니다. 기하학이 주어진 레벨에서 네 개 이상의 격자 셀을 교차했을 때는, 다음의 보다 높은 레벨로 승격됩니다. 그러므로, 세 개의 격자 레벨 10.0e0, 100.0e0 및 1000.0e0이 있는 공간 색인의 경우에는 DB2 Spatial Extender가 각 기하학을 레벨 10.0e0 격자로 우선 교차합니다. 기하학이 네 개 이상의 10.0e0 격자 셀과 교차하면, 그 기하학은 승격되어 레벨 100.0e0 격자와 교차됩니다. 네 개 이상의 교차 결과가 100.0e0 레벨에 있으면, 기하학이

1000.0e0 레벨로 승격됩니다. 1000.0e0 레벨이 가능한 가장 높은 레벨이기 때문에 이 레벨에서는 교차가 공간 색인에 입력되어야 합니다.

그림7에서는 서로 다른 유형의 네 가지 기하학들이 10.0e 격자를 교차하는 방법에 대해 설명합니다. 네 가지 기하학에 대한 23개의 모든 교차가 공간 색인에 기록됩니다.

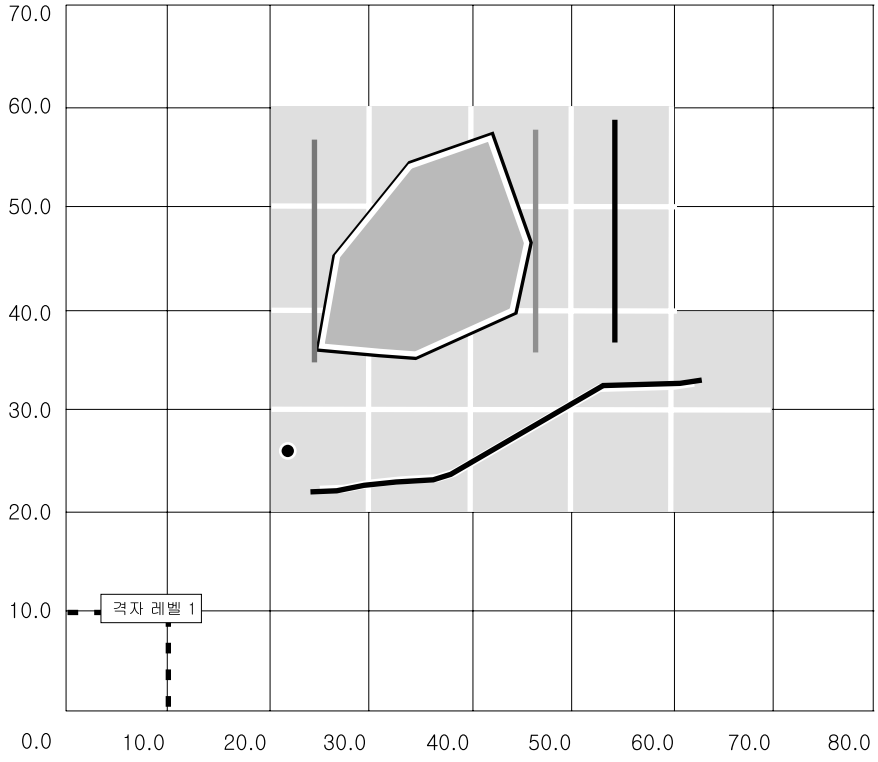


그림 7. 10.0e0 격자 레벨의 응용프로그램

표38에서는 기하학과 그의 해당 격자 교차들을 나열합니다. 네 가지 서로 다른 기하학 유형의 외피들이 10.0e 격자를 교차합니다. 기하학이 교차하는 각 격자 셀의 최소 X 및 Y 좌표가 공간 색인에 입력됩니다.

표 38. 예제 기하학에 대한 10.0e0 격자 셀 항목

기하학	격자 X	격자 Y
다각형	20.0	30.0
다각형	30.0	30.0

표 38. 예제 기하학에 대한 10.0e0 격자 셀 항목 (계속)

기하학	격자 X	격자 Y
다각형	40.0	30.0
다각형	20.0	40.0
다각형	30.0	40.0
다각형	40.0	40.0
다각형	20.0	50.0
다각형	30.0	50.0
다각형	40.0	50.0
수직 선스트링	50.0	30.0
수직 선스트링	50.0	40.0
수직 선스트링	50.0	50.0
점	20.0	20.0
수평 선스트링	20.0	20.0
수평 선스트링	30.0	20.0
수평 선스트링	40.0	20.0
수평 선스트링	50.0	20.0
수평 선스트링	60.0	20.0
수평 선스트링	20.0	30.0
수평 선스트링	30.0	30.0
수평 선스트링	40.0	30.0
수평 선스트링	50.0	30.0
수평 선스트링	60.0	30.0

128 페이지의 그림8에서는 격자 레벨 30.0e0과 60.0e0을 추가하여 교차수를 8개로 크게 줄이는 방법을 보여 줍니다. 이러한 경우에, 기하학 1로 식별된 다각형은 격자 레벨 30.0e0으로 승격되고 기하학 4로 식별된 선스트링은 격자 레벨 60.0e0으로 승격됩니다. 기하학이 10.0e0 레벨에서 가진 9개와 10개의 교차 대신에 승격 뒤에는 두 개의 교차만 가집니다.

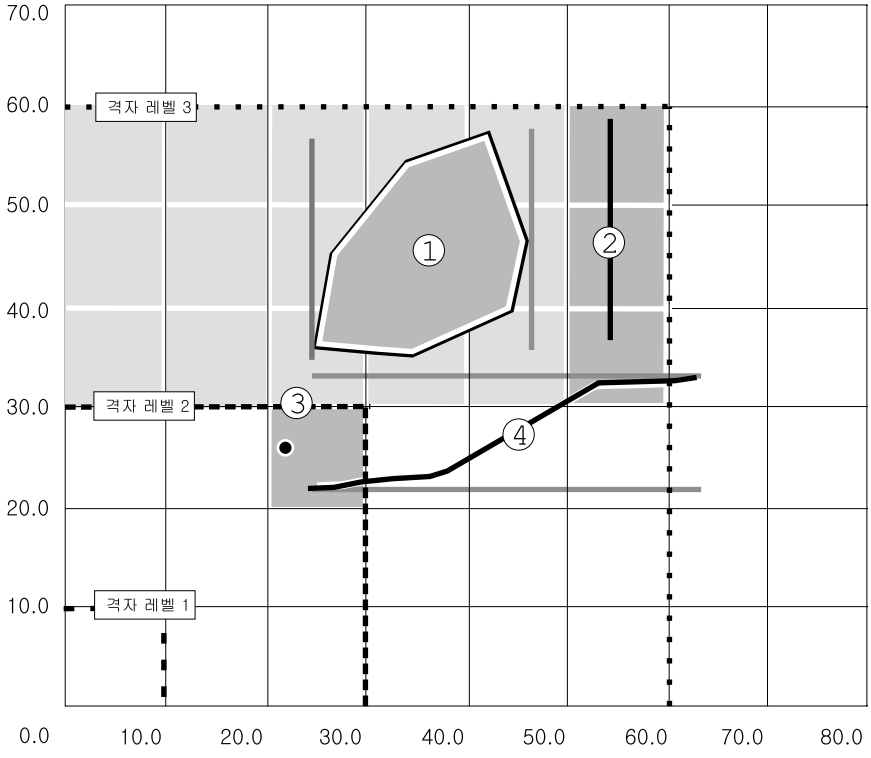


그림 8. 격자 레벨 30.0e0과 60.0e0을 추가한 효과. 기하학 1로 식별된 다각형의 외피는 9개의 격자 셀을 교차합니다. 기하학 2로 식별된 수직 선스트링의 외피는 세 개의 격자 셀을 교차합니다. 기하학 3으로 식별된 점의 외피는 하나의 격자 셀만 교차합니다. 기하학 4로 식별된 선스트링의 외피는 10 개의 격자 셀을 교차합니다.

DB2 Spatial Extender는 CREATE INDEX문에 지정된 격자 레벨 매개변수를 취한 뒤 각 공간 오브젝트를 점검하여 오브젝트가 존재하는 격자 블록의 수와 좌표를 판별합니다. 그림8에서 격자 레벨 10.0e0, 30.0e0 및 60.0e0은 항상 증가하는 선 중량과 서로 다른 명암의 회색으로 표시됩니다. 수직 선스트링과 점 외피 셀 교차점은 10.0e0 격자 레벨에 있는 색인에 입력되는데 이는 둘 다 세 개 이하의 교차점을 생성하기 때문입니다. 다각형은 9개의 10.0e0 격자 셀을 교차하므로, 30.0e0 격자 레벨로 승격됩니다. 이 레벨에서 다각형은 두 개의 격자 셀을 교차하며 이는 색인으로 입력됩니다. 기하학 4로 식별된 선스트링은 10개의 10.0e0 격자 셀을 교차하므로 30.0e0 격자 레벨로 승격됩니다. 이 레벨에서는 이것이 6개의 격자 셀을 교차하므로 다시 60.0e0 격자 레벨로 승격되어 두 개의 교차점을 생성합니다. 그런 다음, 선스트링 60.0e0 격자 교차가 색인으로 입력됩니다. 선스트링이 이 레벨에서 네 개 이상 교차점을 생성한 경우에는, 이 레벨이 기하학이 승격될 수 있는 가장 높은 레벨이기 때문에 그대로 색인에 입력됩니다.

표 39. 3단 색인으로 된 기하학 교차

기하학	격자 X	격자 Y
레벨 1(10.0e0 격자 크기)의 수직 선스트링과 점 사이에 교차		
2	50.0	30.0
2	50.0	40.0
2	50.0	50.0
3	20.0	20.0
레벨 2(30.0e0 격자 크기)의 다각형 교차		
1	0.0	30.0
1	30.0	30.0
레벨 3(60.0e0 격자 크기)의 선스트링 교차		
4	0.0	0.0
4	60.0	0.0

DB2 Spatial Extender는 실제로 어떤 종류의 다각형 격자 구조를 작성하지 않습니다. DB2 Spatial Extender는 컬럼 공간 참조 시스템의 X,Y 옵셋에 원점을 정의하여 각 격자 레벨을 parametrically 명시합니다. 그런 다음, 격자를 양의 좌표 공간으로 확대합니다. parametric 격자를 사용하여 DB2 Spatial Extender는 교차를 수학적으로 생성합니다.

공간 색인 사용 지침

DB2 Spatial Extender는 공간 조회의 성능을 개선시키기 위해 공간 색인에 대해 작업합니다. 가장 기본적이면서 아마도 가장 대중적 공간 조회인 상자 조회를 고려해 봅시다. 이 조회는 사용자 정의 상자 내의 전체 또는 일부인 모든 기하학을 리턴하도록 DB2 Spatial Extender에 지시합니다. 색인이 존재하지 않으면 DB2 Spatial Extender는 모든 기하학을 상자와 비교해야 합니다. 그러나, DB2 Spatial Extender는 왼쪽 하단 좌표가 상자의 좌표보다 크거나 같고 오른쪽 상단 좌표가 상자의 좌표보다 작거나 같은 모든 색인 항목을 찾을 수 있습니다. 좌표 시스템은 색인을 순서화하기 때문에 DB2 Spatial Extender는 후보 기하학의 목록을 신속하게 얻을 수 있습니다. 방금 설명한 프로세스를 *첫번째 패스*라고 합니다.

*두 번째 패스*는 각 후보의 외피가 상자를 교차하는지 판별합니다. 기하학의 격자 셀 외피는 상자와 교차하기 때문에 첫번째 패스에 대해 자격을 갖춘 기하학에 자격을 갖추지 않은 외피가 있을 수도 있습니다.

*세 번째 패스*는 기하학의 어느 부분이 실제적으로 상자 내에 있는지 판별하기 위해 후보의 실제 좌표를 상자와 비교합니다. 마지막이면서 보다 복합적인 비교 프로세스는 총 데이터 상주의 부분 집합으로 구성된 후보 목록에 대해 작업하며 이는 첫번째 두 패스에 의해 상당 부분 줄어 들었습니다.

모든 공간 조회에서는 EnvelopesIntersect 함수 경우를 제외하고 세 개의 패스를 수행합니다. 이 함수는 첫번째 두 패스만 수행합니다. EnvelopesIntersect 함수는 자신 소유의 내장 클리핑 루틴을 자주 사용하고 세 번째 패스의 조절 간격이 필요 없는 화면표시 조작용으로 설계되었습니다.

격자 셀 크기 선택

기하학 외피의 비정규적인 형상은 격자 셀 *크기*의 선택을 복잡하게 만듭니다. 이러한 불규칙성으로 인해 일부 기하학 외피들은 여러 개의 격자를 교차하고 다른 기하학 외피들은 단일 격자 셀 내부에 들어 갑니다. 반대로, 데이터의 공간 분산에 따라 일부 격자 셀들은 여러 기하학 외피들을 교차시킵니다.

공간 색인이 잘 기능하도록 하려면 격자의 수와 *크기*를 적절하게 선택해야 합니다. 일정치 않은 *크기*의 기하학이 들어 있는 공간 컬럼을 고려해 봅시다. 이러한 경우에는 단일 격자 레벨이면 충분합니다. 평균적인 기하학 외피를 둘러싸는 격자 셀 *크기*로 시작하십시오. 응용프로그램을 테스트하는 동안에도, 격자 셀 *크기*를 증가시키면 조회 성능이 개선된다는 것을 알아낼 수도 있습니다. 이는 각 격자 셀에 기하학이 추가로 들어 있기 때문이며, 첫번째 패스에서 자격이 없는 기하학을 보다 빨리 버릴 수 있습니다. 그러나, 계속해서 셀 *크기*를 증가시키면 어느 시점에서 성능이 떨어지기 시작한다는 것을 알게 됩니다. 이는 결국 두 번째 패스에서 보다 많은 후보들이 경합해야 하기 때문입니다.

레벨수 선택

색인화하려는 오브젝트의 상대 *크기*가 동일한 경우에는 사용자가 단일 격자 레벨을 사용할 수 있습니다. 이것이 사실일지라도 모든 컬럼에 동일한 상대 *크기*의 기하학이 들어 있지는 않습니다. 일반적으로 공간 컬럼의 기하학은 여러 *크기*의 간격으로 그룹화될 수 있습니다. 예를 들어, 기하학이 거리, 주요 도로 및 고속도로로 나눠지는 도로 네트워크를 고려해 봅시다. 거리는 거의 모두 동일한 길이를 가지며 한 *크기*의 구간으로 그룹화될 수 있습니다. 이는 주요 도로 및 고속도로에도

마찬가지입니다. 그러므로, 한 *크기* 구간을 나타내는 거리는 첫번째 격자 레벨로 그룹화되고 도로 네트워크는 두 번째, 주요 고속도로는 세 번째 격자 레벨로 그룹화 될 수 있습니다. 또다른 예에는 보다 넓은 시골 지역으로 둘러싸인 작은 도시 지역의 클러스터들이 들어 있는 도 토지 구획 컬럼이 포함됩니다. 이러한 인스턴스에는 작은 도시 지역의 경우에 하나 그리고 보다 넓은 시골 지역의 경우에 또하나씩 하여 두 개의 *크기* 구간과 두 개의 격자레벨이 있습니다. 이러한 상황은 매우 흔하며 이 때는 다중 레벨 격자를 사용해야 합니다.

각 격자 레벨의 셀 *크기*를 선택하려면, 각 *크기* 구간 보다 조금 큰 격자 셀 *크기*를 선택하십시오. 공간 컬럼에 대한 조회를 수행하여 색인을 테스트하십시오.

각 추가 레벨에는 별도의 색인 스캔이 필요합니다. 격자 *크기*를 조금씩 위아래로 조정하면서 성능이 다소 개선되는지 판별하십시오.

제13장 기하학 및 관련 공간 함수

이 장에서는 *기하학*이라는 정보 단위에 대해 토론하는데 이는 좌표로 구성되고 지리적 지형을 기호화합니다. 또한, 입력으로 기하학을 취하고 지리적 지형을 분석하고 지리 정보 시스템 사이에 공간 데이터를 이동하는 데 도움이 되는 결과를 리턴합니다. 다루게 되는 주제는 다음과 같습니다.

- 기하학 특성
- 기하학 등록 정보 : 이 등록 정보와 관련있는 정보를 리턴하는 함수
- 인스턴스화 가능 기하학 : 기하학에 대해 작업하는 함수들
- 다음과 같은 함수
  - 지리적 지형 사이의 관계 및 비교를 보여 주는 함수
  - 기하학을 생성하는 함수
  - 가져오기 가능하고 내보내기 가능한 형식으로 기하학값을 변환하는 함수

기하학 관련 정보

옥스포트 사전에서는 *기하학*을 『선, 각도, 표면 및 입체의 등록 정보 및 관계를 다루는 수학의 분기』로서 정의합니다. 1997년 8월 11일에 Open GIS Consortium Inc.(OGC)에서 출판한 *Open GIS Features for ODBC (SQL) Implementation Specification*에서는 용어에 대한 또다른 정의를 만들어 냈습니다. 과거 수천년 동안 지도 제작자들이 세계 지도를 만드는 데 사용한 기하학 지형을 나타내는 데 단어 *기하학*이 선택되었습니다. 이 새로운 의미의 기하학에 대한 매우 추상적인 정의는 『지면의 지형을 기호화하는 점 또는 점의 총계』입니다.

DB2 Spatial Extender에서 기하학의 *조작 가능한* 정의가 『지리적 지형의 모델』일 수도 있습니다. 모델은 지형 좌표란 용어로 그리고 어떤 경우에는 가시적 기호라는 용어로 표현될 수 있습니다. 모델은 정보를 전달합니다. 예를 들어, 좌표는 고정된 참조 점에 대한 지형 위치를 나타내며 기호는 그 양식의 윤곽을 나타냅니

다. 또한, 모델은 정보를 생성할 수도 있습니다. 예를 들어, ST_Overlaps 함수는 두 개의 가장 가까운 영역의 좌표를 입력으로 취하고, 영역이 겹치는지 여부에 관한 정보를 리턴할 수 있습니다.

기하학이 기호화하는 지형 좌표를 기하학의 등록 *정보*로 간주합니다. 여러 종류의 기하학에는 다음과 같은 기타 다른 등록 정보들도 있습니다.

- *내부*(interior)는 기하학이 기호화하는 지형의 내용을 나타냅니다.
- *외부*(exterior)는 지형 주변의 공간을 나타냅니다.
- *경계*(boundary)는 내용이 끝나고 주변 공간이 시작하는 구분선을 나타냅니다.

136 페이지의 『기하학 및 관련 함수의 등록 정보』에서 이 등록 정보 및 추가 등록 정보에 대한 자세한 내용을 참조하십시오.

DB2 Spatial Extender에 의해 지원되는 기하학은 135 페이지의 그림9에 표시된 대로 계층 구조를 형성합니다. 계층 구조의 6개 구성원은 인스턴스화 가능합니다. 이들은 가시적 기호로 표시될 수 있으며 그림에도 표시됩니다.

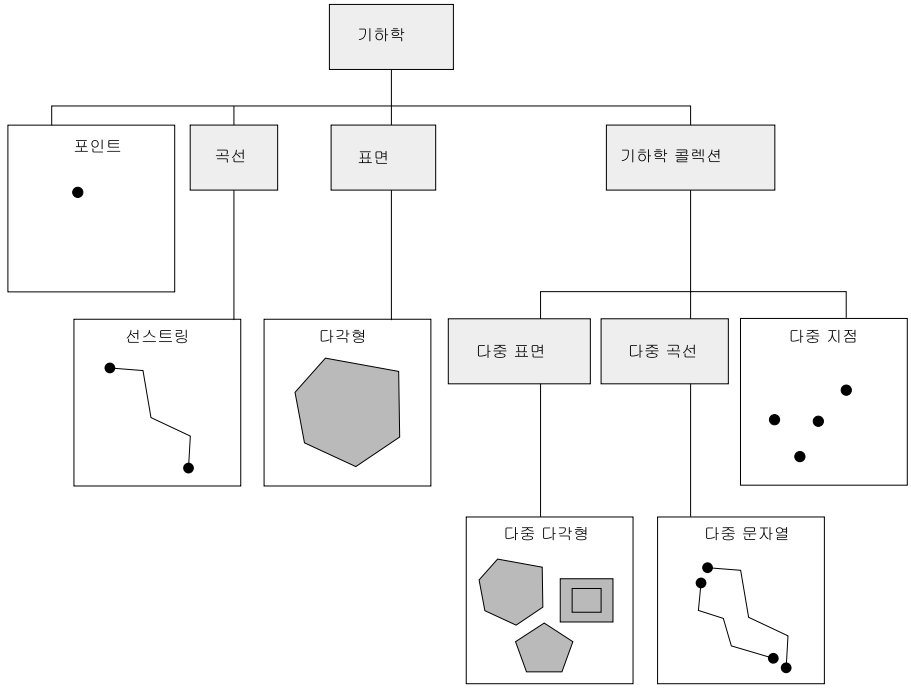


그림 9. DB2 Spatial Extender가 지원하는 기하학 계층 구조. 인스턴스화 가능 기하학은 가시적 기호로 표현될 수 있습니다. 이 기호는 이 기하학의 이름 밑에 표시됩니다.

그림9에 나타난대로 *기하학*이라는 슈퍼클래스는 계층 구조의 루트입니다. 슈퍼클래스는 기본 기하학 서브클래스 및 동종 콜렉션 서브클래스라는 두 개의 범주로 나눠집니다. 기본 기하학에는 다음이 포함됩니다.

- *점* - 이는 동서 좌표선(예: 위선)이 남북 좌표 선(예: 자오선)과 교차하는 위치를 차지할 때 인식되는 이산 지형을 기호화합니다. 예를 들어, 대규모 지도의 표기법에서는 지도상의 각 도시가 위선과 자오선의 교차점에 놓여 있음을 알 수 있습니다. 이러한 스케일에서는 각 도시가 점으로 기호화될 수 있습니다.
- *선스트링* - 선형 지리 대상물(예: 거리, 운하 및 송수관)을 기호화합니다.
- *다각형* - 여러 측면의 지리 대상물(예: 복지 구역, 숲 및 야생 동물 서식지)을 기호화합니다.

동종 콜렉션에는 다음이 포함됩니다.

- *다중점* - 이는 구성요소가 동서 좌표선과 남북 좌표선의 교차점(예: 각 구성원이 위선과 자오선의 교차점에 놓여 있는 섬 군도)에 각각 놓여 있는 다중 파트 지형을 기호화합니다.

- *다중 선스트링* - 이는 선형 단위 또는 구성요소(예: 강 체계 및 고속도로 체계)로 구성된 다중 파트 지형을 기호화합니다.
- *다중 다각형* - 이는 여러 면으로 된 단위 또는 구성요소(예: 특정 영역의 집단 농장 또는 호수 체계)로 이루어진 다중 파트 구성요소를 기호화합니다.

그의 이름이 뜻하는 대로 동종 콜렉션이 기본 기하학의 콜렉션입니다. 기본 기하학 등록 정보를 공유하는 것 이외에도 동종 콜렉션에는 그 자신의 등록 정보도 일부 가지고 있습니다.

135 페이지의 그림9에 표시된 기하학 구현이 DB2 Spatial Extender가 지원하는 공간 데이터 유형입니다. 39 페이지의 『공간 데이터 유형 관련 정보』에서 이 데이터 유형에 대한 자세한 내용을 참조하십시오.

기하학 및 관련 함수의 등록 정보

이 절에서는 기하학 등록 정보 및 이 등록 정보와 관련된 공간 함수를 설명합니다. 절은 다음과 같은 핵심 등록 정보로 시작합니다.

- 기하학이 속한 클래스
- X 및 Y 좌표

이 절에서는 다음에 대해서도 설명합니다.

- Z 좌표
- 치수
- 기하학의 내부, 경계 및 외부
- 단순하거나 단순하지 않은 특성
- 빈 또는 비어 있지 않는 특성
- 기하학의 외피
- 차원
- 기하학 관련 공간 참조 시스템의 식별자

클래스

각 기하학은 135 페이지의 그림9에 표시된 계층 구조 내의 클래스에 속합니다. 133 페이지의 『기하학 관련 정보』에 명시된대로 계층 구조의 6개 서브클래스(점, 선스

트링, 다각형, 다중 다각형, 다중 선스트링 및 다중 다각형)는 인스턴스화 가능합니다. 슈퍼클래스 및 다른 서브클래스들은 인스턴스화 가능하지 않습니다.

ST_GeometryType 함수는 기하학을 취하고 문자열 양식으로 인스턴스화 가능 서브클래스를 리턴합니다. 자세한 내용은 234 페이지의 『ST_GeometryType』을 참조하십시오.

ST_IsValid 함수는 ST_Geometry 데이터 유형에 지정된 기하학을 취합니다. 기하학이 유효하면 함수가 1(TRUE)을 리턴하고 기하학이 유효하지 않으면 0(FALSE)을 리턴합니다. 자세한 내용은 252 페이지의 『ST_IsValid』를 참조하십시오.

X 및 Y 좌표

*X 좌표값*은 동쪽 또는 서쪽 방향으로 참조 점에 상대적인 위치를 나타냅니다. *Y 좌표값*은 북쪽 또는 남쪽 방향으로 참조 점에 상대적인 위치를 나타냅니다. 6 페이지의 『공간 데이터 특성』 및 29 페이지의 『좌표 및 공간 참조 시스템 관련 정보』에서 자세한 내용을 참조하십시오.

Z 좌표

일부 기하학에는 관련된 높이 또는 깊이가 있습니다. 지형의 기하학을 형성하는 각 점에는 지표면에 수직적인 높이나 깊이를 나타내는 선택적인 Z 좌표가 포함될 수 있습니다.

Is3d 술어 함수는 기하학을 취하고, 함수에 Z 좌표가 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. 자세한 내용은 179 페이지의 『Is3d』를 참조하십시오.

치수

치수는 지리적 지형에 관한 정보를 전달하고 지형의 위치를 정의하는 좌표와 함께 저장되는 값입니다. 예를 들어, GIS에서 운송 시스템을 나타낸다고 가정합시다. 응용프로그램에서 직선 거리 또는 이정표를 나타내는 값을 처리하려는 경우에는 시스템의 위치를 정의하는 좌표와 함께 이 값을 저장할 수 있습니다. 치수는 배정밀도 숫자로 저장됩니다.

IsMeasured 술어는 기하학을 취하고, 함수에 치수가 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. 자세한 내용은 180 페이지의 『IsMeasured』를 참조하십시오.

내부, 경계 및 외부

모든 기하학은 내부, 경계 및 외부에 의해 정의된 공간에 위치를 차지합니다. 기하학의 외부란 기하학이 점유하지 않은 모든 공간을 말합니다. 기하학의 경계는 내부와 외부 사이의 인터페이스로서 작용합니다. 내부란 기하학이 점유한 공간을 말합니다. 서브클래스들은 내부 및 외부 등록 정보를 직접 계승하지만, 경계 등록 정보는 각각에 대해 서로 다릅니다.

ST_Boundary 함수는 기하학을 취하고 소스 기하학의 경계를 나타내는 기하학을 리턴합니다. 자세한 내용은 203 페이지의 『ST_Boundary』를 참조하십시오.

단순 또는 비단순

기하학의 일부 서브클래스들(선스트링, 다중점 및 다중 선스트링)은 단순 또는 비단순형입니다. 서브클래스에 부과된 모든 지형학적 규칙을 따르면 그 서브클래스는 단순한 것이고 규칙을 따르지 않으면 그 서브클래스는 단순하지 않은 것입니다. 선스트링이 그의 내부와 교차하지 않으면 이 선스트링은 단순합니다. 다중점의 어떤 요소도 동일한 좌표 공간을 점유하지 않으면 그 다중점은 단순합니다. 다중 선스트링 요소의 어떤 내부도 그 자신의 내부와 교차하지 않으면 그 다중 선스트링은 단순합니다.

ST_IsSimple 술어 함수는 기하학을 취하고, 기하학이 단순하면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. 자세한 내용은 250 페이지의 『ST_IsSimple』를 참조하십시오.

빈 또는 비지 않음

기하학에 아무 점도 없으면 그 기하학은 비어 있습니다. 빈 기하학의 외피, 경계, 내부 및 외부는 NULL입니다. 빈 기하학은 항상 단순하며 Z 좌표 또는 치수를 가질 수 있습니다. 빈 선스트링 및 다중 선스트링의 길이는 0입니다. 빈 다각형 및 다중 다각형의 면적은 0입니다.

ST_IsEmpty 술어 함수는 기하학을 취하고, 기하학이 없으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. 자세한 내용은 246 페이지의 『ST_IsEmpty』를 참조하십시오.

외피

기하학의 외피는 최소 및 최대 (X,Y) 좌표에 의해 형성된 바운딩 기하학입니다. 다음과 같은 예외로 대부분 기하학의 외피는 경계 직사각형을 형성합니다.

- 점의 외피는 점 그 자체이며 이는 최소 및 최대 좌표가 동일하기 때문입니다.
- 수평 또는 수직 선스트링의 외피는 소스 선스트링의 경계(끝점)에 의해 표현되는 선스트링입니다.

ST_Envelope 함수는 기하학을 취하고 그의 외피를 나타내는 바운딩 기하학을 리턴합니다. 자세한 내용은 224 페이지의 『ST_Envelope』을 참조하십시오.

차원

기하학에는 0, 1 또는 2 차원이 있을 수 있습니다. 차원은 다음과 같이 나열됩니다.

0 길이도 없고 영역도 없습니다.

1 길이가 있습니다.

2 영역을 포함합니다.

점 및 다중점 서브클래스는 0차원을 가집니다. 점은 단일 좌표로 모델화될 수 있는 차원적인 지형을 나타내는 반면에, 다중점 서브클래스는 단절된 좌표의 클러스터로 모델화되어야 하는 데이터를 나타냅니다.

서브클래스 선스트링 및 다중 선스트링은 1차원을 가집니다. 이는 도로 세그먼트, 분기 강 체계 및 자연적으로 직선 형태인 기타 다른 지형들을 저장합니다.

다각형 및 다중 다각형 서브클래스는 2차원을 가집니다. 주계가 숲, 토지 필지 및 수계 본체와 같이 정의가능한 영역을 에워싸는 지형은 다각형 또는 다중 다각형 데이터 유형으로 표현될 수 있습니다.

차원은 서브클래스의 등록 정보로서 뿐만 아니라 두 지형의 공간적인 관계를 판별하는 데 중요한 요소로서도 중요합니다. 결과 지형(들)의 차원으로 조작의 성공 여부를 판별합니다. DB2 Spatial Extender는 지형의 차원을 조사하여 지형을 비교할 방법을 판별합니다.

ST_Dimension 함수는 기하학을 취하고 그의 차원을 정수로 리턴합니다. 자세한 내용은 217 페이지의 『ST_Dimension』을 참조하십시오.

공간 참조 시스템 식별자

공간 참조 시스템은 각 기하학에 대한 좌표 변환을 식별합니다.

데이터베이스에 알려진 모든 공간 참조 시스템은 DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰를 통해 액세스될 수 있습니다. 119 페이지의 『제11장 카탈로그 뷰』에서 이 뷰에 대한 자세한 내용을 참조하십시오.

ST_SRID 함수는 기하학을 취하고 그의 공간 참조 식별자를 정수로 리턴합니다. 자세한 내용은 289 페이지의 『ST_SRID』를 참조하십시오.

ST_Transform 함수는 기하학이 현재 지정된 공간 참조 시스템이 아닌 다른 공간 참조 시스템에 기하학을 지정합니다. 자세한 내용은 295 페이지의 『ST_Transform』을 참조하십시오.

인스턴스화 가능 기하학 및 관련 함수

이 절에서는 인스턴스화 가능 기하학의 6개 서브클래스에 대한 윤곽을 잡으며 그 서브클래스에 대해 작업하는 함수들을 설명합니다. 서브클래스는 다음과 같습니다.

- 점
- 선스트링
- 다각형
- 다중점
- 다중 선스트링
- 다중 다각형

이 서브클래스가 속해 있는 계층 구조 및 서브클래스와 연관된 가시적 기호에 대해서는 135 페이지의 그림9를 참조하십시오.

점

점은 좌표 공간에서 단일 위치를 차지하는 0차원 기하학입니다. 점은 이 위치를 정의하는 X 좌표와 Y 좌표로 구성됩니다. 여기에는 Z 좌표와 치수도 포함될 수 있습니다.

점은 단순하며 NULL 경계값을 가집니다. 점은 유정, 이정표 및 해발과 같은 지형을 정의하는 데 자주 사용됩니다.

점 서브클래스에 대해 단독으로 작용하는 함수는 다음과 같습니다.

ST_Point

X 좌표, 관련 Y 좌표 및 이 좌표들이 속해 있는 공간 참조 시스템의 식별자를 취하고 좌표가 정의하는 점을 리턴합니다. 자세한 내용은 279 페이지의 『ST_Point』를 참조하십시오.

ST_CoordDim

점에 들어 있는 좌표를 나타내는 값과 치수가 들어 있는지 여부도 리턴합니다. 이 값을 *좌표 차원*이라고 합니다. 가능한 좌표 차원은 다음과 같습니다.

2 점은 X 좌표와 Y 좌표로 구성됩니다.

3 점은 X 좌표, Y 좌표 및 Z 좌표로 구성됩니다.

4 점은 X 좌표, Y 좌표, Z 좌표 및 치수로 구성됩니다.

자세한 내용은 212 페이지의 『ST_CoordDim』을 참조하십시오.

ST_PointFromText

점의 OGC 잘 알려진 텍스트 (WKT) 표현식을 취하고 점을 리턴합니다. 자세한 내용은 276 페이지의 『ST_PointFromText』를 참조하십시오.

ST_X ST_Point 데이터 유형의 X 좌표값을 배정밀도 숫자로 리턴합니다. 자세한 내용은 303 페이지의 『ST_X』를 참조하십시오.

ST_Y ST_Point 데이터 유형의 Y 좌표값을 배정밀도 숫자로 리턴합니다. 자세한 내용은 304 페이지의 『ST_Y』를 참조하십시오.

Z ST_Point 데이터 유형의 Z 좌표값을 배정밀도 숫자로 리턴합니다. 자세한 내용은 305 페이지의 『Z』를 참조하십시오.

M ST_Point 데이터 유형의 치수를 배정밀도 숫자로 리턴합니다. 자세한 내용은 187 페이지의 『M』를 참조하십시오.

선스트링

선스트링은 중간에 삽입된 선형 경로를 정의하는 점 연속물로 저장된 1차원 오브젝트입니다. 선스트링이 그의 내부와 교차하지 않으면 이 선스트링은 단순합니다. 닫힌 선스트링의 끝점(경계)은 공간에서 동일한 점을 점유합니다. 선스트링이 닫혀 있고 그의 내부가 그 자신과 교차하지 않으면 그 선스트링은 링입니다. 수퍼클래스 기하학으로부터 계승된 다른 등록 정보외에도 선스트링은 길이를 갖습니다. 선스트링은 도로, 강 및 전선과 같은 선형 지형을 정의하는 데에도 사용됩니다.

시작점과 끝점이 동일한 단순한 선스트링을 *링*이라고 합니다.

경계가 NULL인 경우에 선스트링이 닫히지 않는한 일반적으로 끝점이 선스트링의 경계를 형성합니다. 선스트링의 내부는 내부가 연속적인 경우에 선스트링이 닫히지 않는 한 끝점들 사이에 놓여 있는 연결된 경로입니다.

선스트링에 대해 수행되는 함수는 다음과 같습니다.

ST_StartPoint

선스트링을 취하고 이의 첫번째 점을 리턴합니다. 자세한 내용은 290 페이지의 『ST_StartPoint』를 참조하십시오.

ST_EndPoint

선스트링을 취하고 이의 최종 점을 리턴합니다. 자세한 내용은 222 페이지의 『ST_Endpoint』를 참조하십시오.

ST_PointN

선스트링과 *n* 번째 점에 대한 색인을 취하고 그 점을 리턴합니다. 자세한 내용은 280 페이지의 『ST_PointN』을 참조하십시오.

ST_Length

선스트링을 취하고 이의 길이를 배정밀도 숫자로 리턴합니다. 자세한 내용은 254 페이지의 『ST_Length』를 참조하십시오.

ST_NumPoints

선스트링을 취하고 이의 점 수를 정수로 연속해서 리턴합니다. 자세한 내용은 270 페이지의 『ST_NumPoints』를 참조하십시오.

ST_IsRing

선스트링을 취하고, 선스트링이 링이면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. 자세한 내용은 248 페이지의 『ST_IsRing』을 참조하십시오.

ST_IsClosed

선스트링을 취하고, 선스트링이 닫혀 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. 자세한 내용은 244 페이지의 『ST_IsClosed』를 참조하십시오.

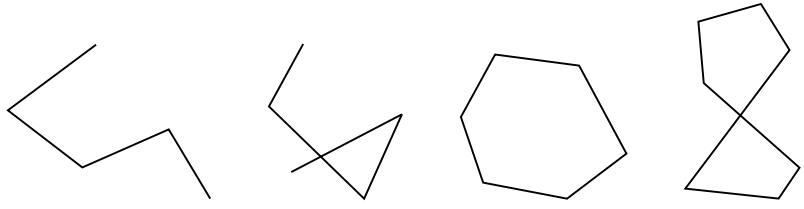


그림 10. 선스트링 오브젝트

1. 닫히지 않은 단순한 선스트링.
2. 닫히지 않고 단순하지 않은 선스트링.
3. 닫혀 있는 단순한 선스트링이면서 링.
4. 닫혀 있는 단순하지 않은 선스트링. 링은 아님.

다각형

다각형은 그의 외부 바운딩 링과 0개 이상의 내부 링을 정의하는 점 연속물로 저장된 2차원 표면입니다. 다각형의 링은 겹쳐질 수 없습니다. 그러므로, 정의상으로 다각형은 항상 단순합니다. 거의 대부분 다각형은 대지, 수역 본체의 일부분과 공간 extent를 가진 다른 지형을 정의합니다.

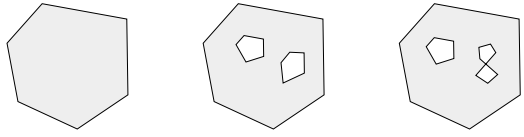


그림 11. 다각형.

1. 경계가 외부 링으로 정의된 다각형.
2. 경계가 하나의 외부 링과 두 개의 내부 링으로 정의된 다각형. 내부 링 안에 있는 영역은 다각형 외부의 일부분입니다.
3. 링이 단일 정접 점에서 교차하기 때문에 적법한 다각형입니다.

외부 및 내부 링은 다각형의 경계를 정의하고 링 사이에 묶인 공간은 다각형의 내부를 정의합니다. 다각형의 링은 정접 점에서 교차할 수는 있지만 절대 관통할 수는 없습니다. 수퍼클래스 기하학으로부터 계승된 다른 등록 정보외에도 다각형에는 영역이 있습니다.

다각형에 대해 수행되는 함수는 다음과 같습니다.

ST_Area

다각형을 취하고 이의 영역을 배정밀도 숫자로 리턴합니다. 자세한 내용은 199 페이지의 『ST_Area』를 참조하십시오.

ST_ExteriorRing

다각형을 취하고 이의 외부 링을 선스트링으로 리턴합니다. 자세한 내용은 227 페이지의 『ST_ExteriorRing』을 참조하십시오.

ST_NumInteriorRing

다각형을 취하고 그 안에 들어 있는 내부 링 수를 리턴합니다. 자세한 내용은 269 페이지의 『ST_NumInteriorRing』을 참조하십시오.

ST_InteriorRingN

다각형과 색인을 취하고 *n* 번째 내부 링을 선스트링으로 리턴합니다. 자세한 내용은 236 페이지의 『ST_InteriorRingN』을 참조하십시오.

ST_Centroid

다각형을 취하고 다각형 extent의 중심이 되는 점을 리턴합니다. 자세한 내용은 207 페이지의 『ST_Centroid』를 참조하십시오.

ST_PointOnSurface

다각형을 취하고 다각형의 표면에 위치하게 되는 점을 리턴합니다. 자세한 내용은 282 페이지의 『ST_PointOnSurface』를 참조하십시오.

ST_Perimeter

다각형을 취하고 그 표면의 주계를 리턴합니다. 자세한 내용은 275 페이지의 『ST_Perimeter』를 참조하십시오.

다중점

다중점은 점의 콜렉션이며 그의 요소와 같이 0차원을 갖습니다. 다중점의 어떤 요소도 동일한 좌표 공간을 점유하지 않으면 그 다중점은 단순합니다. 다중점의 경계는 NULL입니다. 다중점은 항공 방송 패턴 및 전염성 있는 폭동 사건과 같은 현상을 정의하는 데 사용될 수도 있습니다.

다중점에 대해 수행되는 함수는 다음과 같습니다.

ST_NumGeometries

동종 콜렉션을 취하고 그 안에 들어 있는 기본 기하학 요소의 수를 리턴합니다. 자세한 내용은 268 페이지의 『ST_NumGeometries』를 참조하십시오.

ST_GeometryN

동종 콜렉션과 색인을 취하고 n 번째 기본 기하학을 리턴합니다. 자세한 내용은 233 페이지의 『ST_GeometryN』을 참조하십시오.

다중 선스트링

다중 선스트링은 선스트링의 콜렉션입니다. 다중 선스트링이 선스트링 요소의 끝점에서만 교차하면 이 다중 선스트링은 단순합니다. 선스트링 요소의 내부가 교차하면 이 다중 선스트링은 단순하지 않습니다.

다중 선스트링의 경계는 선스트링 요소의 교차되지 않는 끝점입니다. 다중 선스트링의 모든 선스트링 요소가 닫히면 그 다중 선스트링은 닫힙니다. 모든 요소의 모든 끝점이 교차되면 다중 선스트링의 경계는 NULL입니다. 수퍼클래스 기하학으로부터 계승된 다른 등록 정보외에도 선스트링은 길이를 갖습니다. 다중 선스트링은 하천 또는 도로 네트워크를 정의하는 데 사용됩니다.

다중 선스트링에 대해 수행되는 함수는 다음과 같습니다.

ST_Length

다중 선스트링을 취하고 이의 모든 선스트링 요소의 누적 길이를 배정밀도 숫자로 리턴합니다. 자세한 내용은 254 페이지의 『ST_Length』를 참조하십시오.

ST_IsClosed

다중 선스트링을 취하고, 선스트링이 닫혀 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. 자세한 내용은 244 페이지의 『ST_IsClosed』를 참조하십시오.

ST_NumGeometries

동종 콜렉션을 취하고 그 안에 들어 있는 기본 기하학 요소의 수를 리턴합니다. 자세한 내용은 268 페이지의 『ST_NumGeometries』를 참조하십시오.

ST_GeometryN

동종 콜렉션과 색인을 취하고 n 번째 기본 기하학을 리턴합니다. 자세한 내용은 233 페이지의 『ST_GeometryN』을 참조하십시오.

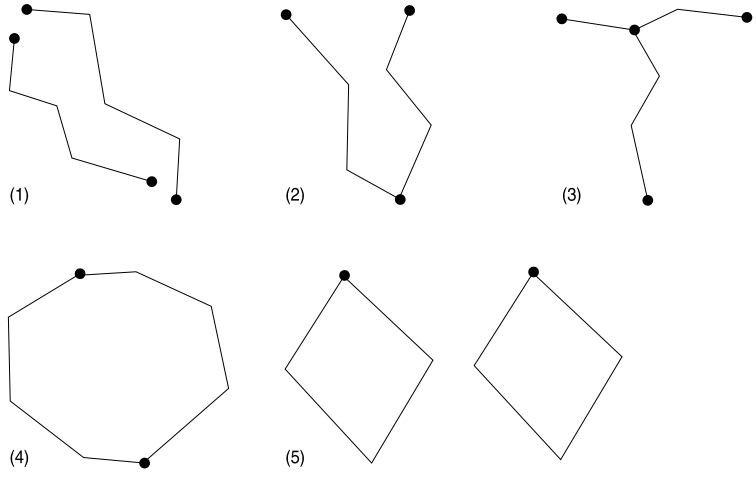


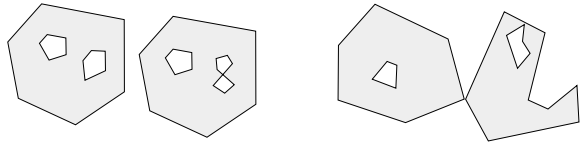
그림 12. 다중 선스트링.

1. 경계가 두 선스트링 요소의 네 끝점으로 정의되는 단순한 다중 선스트링.
2. 선스트링 요소의 끝점만이 교차하기 때문에 단순한 다중 선스트링. 경계는 교차하지 않는 두 끝점에 의해 정의됩니다.
3. 선스트링 요소 중 한 요소의 내부가 교차되기 때문에 단순하지 않은 선스트링. 이 다중 선스트링의경계는 교차점을 포함하여 네 개의 끝점으로 정의됩니다.
4. 닫히지 않은 단순한 다중 선스트링. 요소 선스트링이 닫히지 않기 때문에 이 다중 선스트링은 닫히지 않습니다. 요소 선스트링의 내부 중 어느 것도 교차하지 않기 때문에 단순합니다.
5. 닫힌 상태의 단순한 다중 선스트링. 이 다중 선스트링은 그의 모든 요소가 닫혔기 때문에 닫힌 상태입니다. 이는 어느 요소도 내부에서 교차하지 않기 때문에 단순합니다.

다중 다각형

다중 다각형의 경계는 요소의 외부 및 내부 링의 총 누적 길이입니다. 다중 다각형의 내부는 그 요소 다각형의 누적 내부로 정의됩니다. 다중 다각형 요소의 경계는 접점에서만 교차할 수 있습니다. 수퍼클래스 기하학으로부터 계승된 다른 등록

정보외에도 다중 다각형에는 영역이 있습니다. 다중 다각형은 숲 지층과 같은 지형 또는 섬 군도와 같은 비연속적인 토지 구획 지형을 정의합니다.



(1) (2)

그림 13. 다중 다각형.

1. 두 개의 다각형 요소가 있는 다중 다각형. 경계는 두 개의 외부 링과 세 개의 내부 링에 의해 정의됩니다.
2. 두 개의 다각형 요소가 있는 다중 다각형. 경계는 두 개의 외부 링과 두 개의 내부 링에 의해 정의됩니다. 두 다각형 요소는 접점에서 교차합니다.

다중 다각형에 대해 수행되는 함수는 다음과 같습니다.

ST_Area

다중 다각형을 취하고 이의 다각형 요소의 누적 영역을 배정밀도 숫자로 리턴합니다. 자세한 내용은 199 페이지의 『ST_Area』를 참조하십시오.

ST_Centroid

다중 다각형을 취하고 이의 기하학적 가중 중심지인 점을 리턴합니다. 자세한 내용은 207 페이지의 『ST_Centroid』를 참조하십시오.

ST_NumGeometries

동종 콜렉션을 취하고 그 안에 들어 있는 기본 기하학 요소의 수를 리턴합니다. 자세한 내용은 268 페이지의 『ST_NumGeometries』를 참조하십시오.

ST_GeometryN

동종 콜렉션과 색인을 취하고 n 번째 기본 기하학을 리턴합니다. 자세한 내용은 233 페이지의 『ST_GeometryN』을 참조하십시오.

관계 및 비교를 나타내고 기하학을 생성하며 값의 형식을 변환하는 함수

앞의 절에서는 공간 함수의 세 범주에 대해 소개했습니다.

- 기하학 등록 정보와 관련된 함수
- 특정 기하학과 관련된 함수

이 절에서는 세 개의 범주를 추가로 소개합니다.

- 지리적 지형이 관련되거나 비교하는 방식을 결정하는 함수
- 새 기하학을 생성하는 함수
- 기하학값을 가져오기 또는 내보내기될 수 있는 형식으로 변환하는 함수

지리적 지형 사이의 관계 또는 비교를 나타내는 함수

여러 공간 함수는 지리적 지형이 서로에게 관련짓거나 서로를 비교하는 방법에 대한 정보를 리턴합니다. *술어*라고 하는 이러한 함수의 대부분은 부울 함수입니다. 이 절에서는 일반적으로 술어에 대해 설명한 후 개별적으로 각 함수에 대해 설명합니다.

술어 함수

술어 함수는 비교 결과가 함수 기준을 충족하면 1(TRUE)을 리턴하고 비교에 실패하면 0(FALSE)을 리턴합니다. 공간 관계를 테스트하는 술어는 서로 다른 유형 또는 차원이 될 수 있는 기하학쌍을 비교합니다.

술어는 제출된 기하학의 X 및 Y 좌표를 비교합니다. Z 좌표와 치수(존재할 경우)는 무시됩니다. 그로 인해, Z 좌표 또는 치수가 있는 기하학을 Z 좌표 또는 치수가 없는 기하학과 비교할 수 있는 것입니다.

Dimensionally Extended 9 Intersection Model(DE-9IM)[1]은 서로 다른 유형과 차원의 기하학 사이의 쌍으로 된 공간 관계를 정의하는 수학적 접근 방식입니다. 이

1. DE-91M은 Egenhofer 및 Herring의 9개 교차 모델을 공간적으로 확장한 Clementini와 Felice가 개발했습니다. DE-91M은 네 명의 작성자인 Clementini, Eliseo, Di Felice, van Osstrom의 합작품입니다. 이들은 ″A Small Set of Formal Topological Relationships Suitable for End-User Interaction,″ D. Abel 및 B.C. Ooi (Ed.), *Advances in Spatial Database—Third International Symposium. SSD '93*. LNCS 692. Pp. 277-295에 모델을 발표했습니다. Springer-Verlag Singapore (1993)

모델은 결과로 발생되는 교차의 차원을 고려하여 모든 유형의 기하학 사이의 공간 관계를 그의 내부, 경계 및 외부의 교차쌍으로서 나타냅니다.

주어진 기하학 *a* 및 *b*: I(*a*), B(*a*) 및 E(*a*)는 *a*의 내부, 경계 및 외부를 나타냅니다. 그리고, I(b), B(b) 및 E(b)는 *b*의 내부, 경계 및 외부를 나타냅니다. I(a), B(a), E(a)와 I(b), B(b) 및 E(b)를 교차하여 3 곱하기 3 행렬을 생성합니다. 각각을 교차하면 서로 다른 차원의 기하학이 생성될 수 있습니다. 예를 들어, 두 다각형 경계의 교차는 점과 선스트링으로 구성되는데, 이러한 경우에는 dim 함수가 최대 차원 1을 리턴합니다.

dim 함수는 값 1, 0, 1 또는 2를 리턴합니다. 1은 널 세트 또는 dim(널)에 해당되는데 이는 교차가 일어나지 않을 때 리턴되는 값입니다.

	내부	**경계**	**외부**
내부	dim(I(a) ∩ I(b))	dim(I(a) ∩ B(b))	dim(I(a) ∩ E(b))
경계	dim(B(a) ∩ I(b))	dim(B(a) ∩ B(b))	dim(B(a) ∩ E(b))
외부	dim(E(a) ∩ I(b))	dim(E(a) ∩ B(b))	dim(E(a) ∩ E(b))

공간 관계 술어의 결과는 DE-9IM에 대해 허용가능한 값을 나타내는 패턴 행렬과 술어의 결과를 비교하여 이해하거나 검증할 수 있습니다.

패턴 행렬에는 각각의 교차 행렬 셀에 대한 허용가능한 값이 들어 있습니다. 가능한 패턴값은 다음과 같습니다.

T 교차가 존재해야 함, dim = 0, 1 또는 2.

F 교차가 존재하지 말아야 함, dim = -1.

***** 교차가 존재하는지 여부에 상관하지 않음, dim = -1, 0, 1 또는 2.

0 교차가 존재해야 하고 그의 최대 차원은 0이어야 함, dim = 0.

1 교차가 존재해야 하고 그의 최대 차원은 1이어야 함, dim = 1

2 교차가 존재해야 하고 그의 최대 차원은 2이어야 함, dim = 2

Egenhofer M.J. 및 Herring, J.에 의한 9개 교차 모델은 ″Categorizing binary topological relationships between regions, lines, and points in geographic databases,″ *Tech. Report, Department of Surveying Engineering*, University of Maine, Orono, ME 1991에 발표했습니다.

예를 들어, ST_Within 술어에 대한 다음의 패턴 행렬에는 값 T, F 및 *가 포함됩니다.

표 40. ST_Within에 대한 행렬. 기하학 조합에 대한 ST_Within 술어의 패턴 행렬

		b		
		내부	**경계**	**외부**
a	**내부**	T	*	F
	경계	*	*	F
	외부	*	*	*

ST_Within 술어는 양쪽 기하학의 내부가 교차할 때 그리고 *a*의 내부와 경계가 *b*의 외부와 교차하지 않을 때 TRUE를 리턴합니다. 다른 모든 조건들은 상관없습니다.

각 술어에는 적어도 하나의 패턴 행렬이 있지만, 일부 술어는 다양한 기하학 유형 조합의 관계를 설명하기 위해 둘 이상의 패턴 행렬을 필요로 합니다.

ST_Equals

ST_Equals는 동일한 유형의 두 기하학이 동일한 X,Y 좌표값을 가지면 1(TRUE)을 리턴합니다.

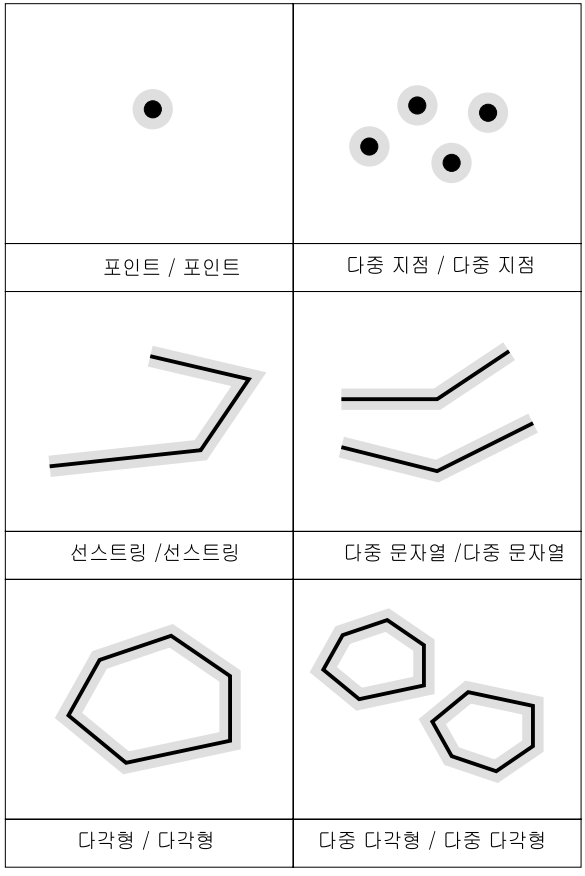


그림 14. ST_Equals. 기하학에 일치하는 X,Y 좌표가 있으면 그 기하학은 동일함.

표 41. 대등성(equality)에 대한 행렬. 대등성에 대한 DE-9IM 패턴 행렬을 통해, 내부가 교차하고 둘 중 한 기하학의 파트 내부 또는 경계 어느 것도 서로의 외부와 교차하지 않는다는 것을 확인할 수 있습니다.

		b		
		내부	**경계**	**외부**
a	**내부**	T	*	F
	경계	*	*	F
	외부	F	F	*

자세한 내용은 226 페이지의 『ST_Equals』을 참조하십시오.

ST_OrderingEquals

ST_OrderingEquals는 두 개의 기하학을 비교하여 기하학이 같으면서 좌표가 동일한 순서도 되어 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. 자세한 내용은 271 페이지의 『ST_OrderingEquals』을 참조하십시오.

ST_Disjoint

ST_Disjoint는 두 기하학의 교차가 빈 세트이면 1(TRUE)을 리턴합니다.

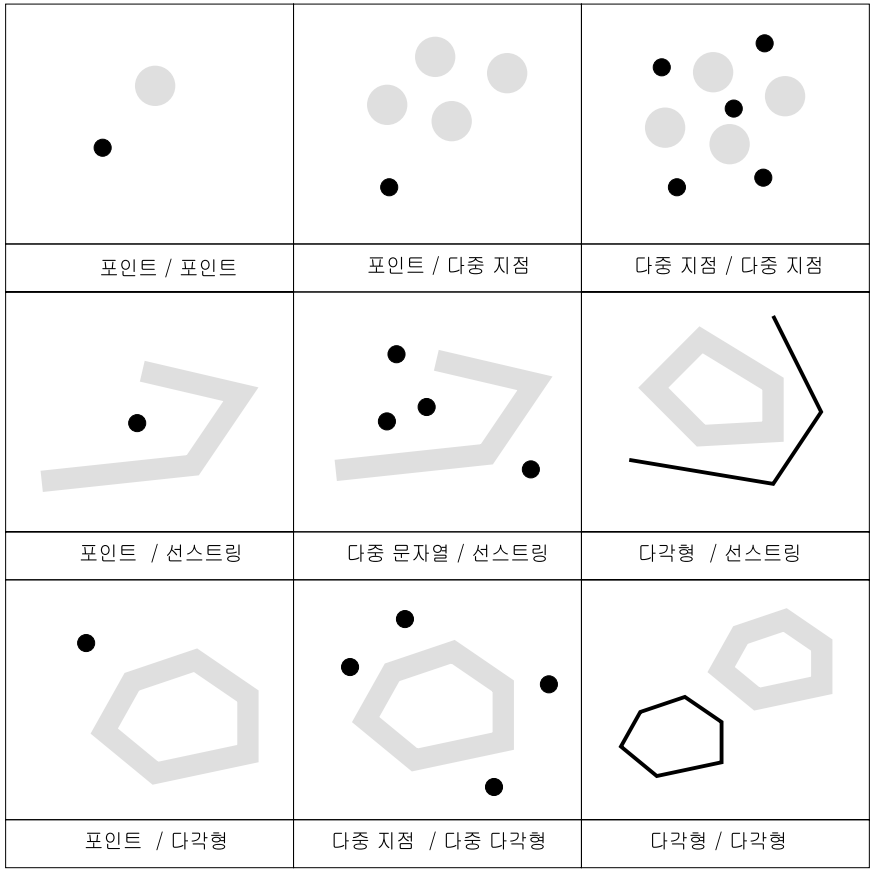


그림 15. ST_Disjoint. 기하학이 어떤 방식으로 서로와 교차하지 않으면 기하학은 결합해제됩니다.

표 42. ST_Disjoint에 대한 행렬. ST_Disjoint 술어의 패턴 행렬은 기하학의 내부나 경계가 교차하지 않음을 간단히 나타냅니다.

		b		
		내부	경계	외부
a	내부	F	F	*
	경계	F	F	*
	외부	*	*	*

자세한 내용은 219 페이지의 『ST_Disjoint』를 참조하십시오.

ST_Intersects

ST_Intersects는 교차한 결과 세트가 비어 있지 않으면 1(TRUE)을 리턴합니다. Intersects는 ST_Disjoint의 정확히 반대 결과를 리턴합니다.

다음과 같은 패턴 행렬의 조건이 TRUE를 리턴하면 ST_Intersects 술어는 TRUE를 리턴합니다.

표 43. ST_Intersects에 대한 행렬(1). ST_Intersects 술어는 두 기하학의 내부가 교차하면 TRUE를 리턴합니다.

		b		
		내부	경계	외부
a	내부	T	*	*
	경계	*	*	*
	외부	*	*	*

표 44. ST_Intersects에 대한 행렬(2). ST_Intersects 술어는 첫번째 기하학의 경계가 두 번째 기하학의 경계와 교차하면 TRUE를 리턴합니다.

		b		
		내부	경계	외부
a	내부	*	T	*
	경계	*	*	*
	외부	*	*	*

표 45. ST_Intersects에 대한 행렬(3). ST_Intersects 술어는 첫번째 기하학의 경계가 두 번째 기하학의 내부와 교차하면 TRUE를 리턴합니다.

		b		
		내부	경계	외부
a	내부	*	*	*
	경계	T	*	*
	외부	*	*	*

표 46. ST_Intersects에 대한 행렬(4). ST_Intersects 술어는 두 기하학 중 하나의 경계가 교차하면 TRUE를 리턴합니다.

		b		
		내부	경계	외부
a	내부	*	*	*
	경계	*	T	*
	외부	*	*	*

자세한 내용은 243 페이지의 『ST_Intersects』를 참조하십시오.

EnvelopesIntersect

이 함수는 두 기하학의 외피가 교차하면 1(TRUE)를 리턴합니다. 이는 ST_Intersects를 효과적으로 구현하는 편리한 함수입니다(`ST_Envelope(g1),ST_Envelope(g2)`). 자세한 내용은 177 페이지의 『EnvelopesIntersect』를 참조하십시오.

ST_Touches

ST_Touches는 두 기하학에 공통적인 점 중 어느 것도 두 기하학의 내부와 교차하지 않으면 1(TRUE)을 리턴합니다. 적어도 하나의 기하학이 선스트링, 다각형, 다중 선스트링 또는 다중 다각형이어야 합니다.

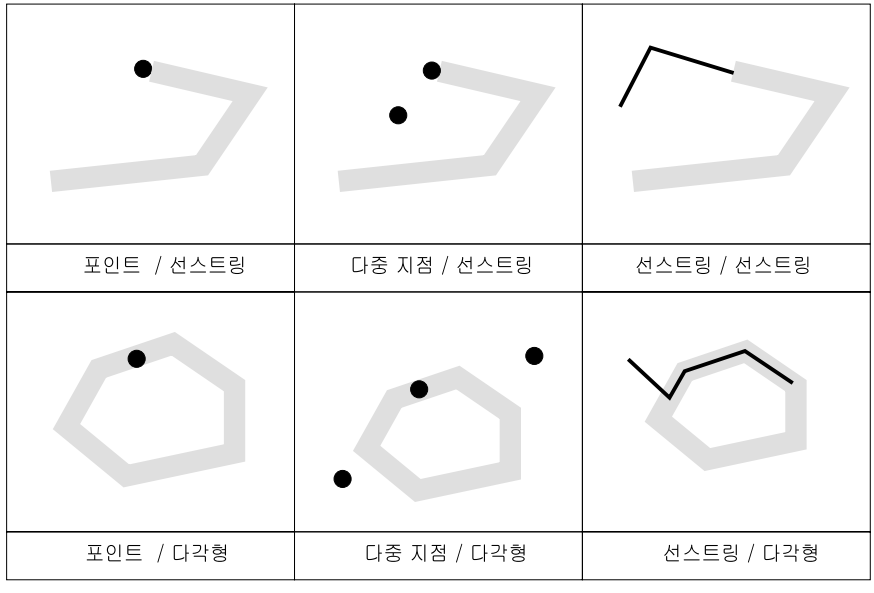


그림 16. ST_Touches

패턴 행렬에서는 기하학의 내부가 교차하고 기하학의 경계가 다른 내부 또는 그의 경계와 교차할 때 ST_Touches 술어가 TRUE를 리턴한다는 점을 보여 줍니다.

표 47. ST_Touches에 대한 행렬(1)

		b		
		내부	경계	외부
a	내부	F	T	*
	경계	*	*	*
	외부	*	*	*

표 48. ST_Touches에 대한 행렬(2)

		b		
		내부	경계	외부
a	내부	F	*	*
	경계	T	*	*
	외부	*	*	*

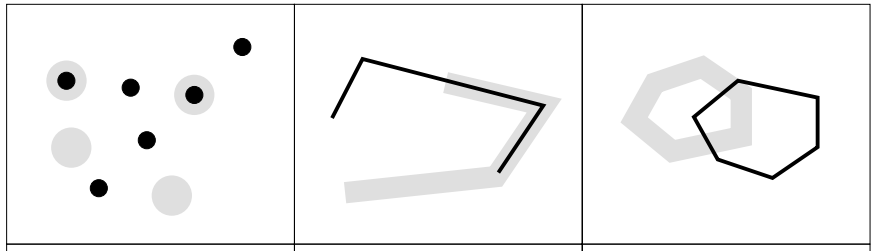
표 49. ST_Touches에 대한 행렬(3)

		b		
		내부	경계	외부
a	내부	F	*	*
	경계	*	T	*
	외부	*	*	*

자세한 내용은 294 페이지의 『ST_Touches』를 참조하십시오.

ST_Overlaps

ST_Overlaps는 동일한 차원의 두 기하학을 비교합니다. 교차 세트 결과가 두 기하학과 다르지만 같은 차원을 가지면 1(TRUE)을 리턴합니다.



다중 지점 / 다중 지점	선스트링 / 선스트링	다각형 / 다각형

그림 17. ST_Overlaps

표50의 패턴 행렬은 다각형/다각형, 다중점/다중점 및 다중 다각형/다중 다각형 오버레이에 적용됩니다. 이러한 조합의 경우에 두 기하학의 내부가 다른 내부 및 외부와 교차하면 오버레이 술어가 TRUE를 리턴합니다.

표 50. ST_Overlaps에 대한 행렬(1)

		b		
		내부	**경계**	**외부**
a	**내부**	T	*	T
	경계	*	*	*
	외부	T	*	*

표51의 패턴 행렬은 선스트링/선스트링 및 다중 선스트링/다중 선스트링 오버레이에 적용됩니다. 이러한 경우의 기하학 교차 결과는 1차원 기하학(다른 선스트링)이어야 합니다. 내부 교차의 차원이 1이면 ST_Overlaps 술어가 FALSE를 리턴하지만 ST_Crosses 술어는 TRUE를 리턴합니다.

표 51. ST_Overlaps에 대한 행렬(2)

		b		
		내부	**경계**	**외부**
a	**내부**	1	*	T
	경계	*	*	*
	외부	T	*	*

자세한 내용은 273 페이지의 『ST_Overlaps』를 참조하십시오.

ST_Crosses

ST_Crosses는 교차 결과가 두 소스 기하학의 최대 차원 보다 차원이 하나 적고 교차 세트가 두 소스 기하학에 대해 내부인 기하학 경우에는 1(TRUE)을 리턴합

니다. ST_Crosses는 다중점/다각형, 다중점/선스트링, 선스트링/선스트링, 선스트링/다각형 및 선스트링/다중 다각형 비교에 대해서만 1(TRUE)을 리턴합니다.

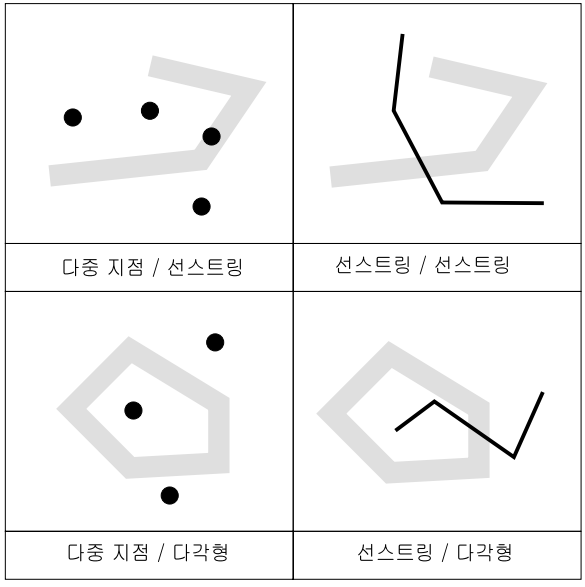


표52의 패턴 행렬은 다중점/선스트링, 다중점/다중 선스트링, 다중점/다각형, 다중점/다중 다각형, 선스트링/다각형, 선스트링/다중 다각형에 적용됩니다. 행렬은 1차(기하학 *a*)의 내부가 보조(기하학 *b*)의 외부와 교차해야 함을 나타냅니다.

표 52. ST_Crosses에 대한 행렬(1)

		b		
		내부	**경계**	**외부**
a	**내부**	T	*	T
	경계	*	*	*
	외부	*	*	*

159 페이지의 표53의 패턴 행렬은 선스트링/선스트링, 선스트링/다중 선스트링 및 다중 선스트링/다중 선스트링에 적용됩니다. 행렬에서는 내부 교차의 차원이 0(점에서 교차)이어야 함을 나타냅니다. 이 교차의 차원이 1이면(선스트링에서 교차) ST_Crosses 술어는 FALSE를 리턴하지만 ST_Overlaps 술어는 TRUE를 리턴합니다.

표 53. ST_Crosses에 대한 행렬(2)

		b		
		내부	**경계**	**외부**
a	**내부**	0	*	*
	경계	*	*	*
	외부	*	*	*

자세한 내용은 214 페이지의 『ST_Crosses』를 참조하십시오.

ST_Within

ST_Within은 첫번째 기하학이 완전하게 두 번째 기하학 내에 있으면 1(TRUE)을 리턴합니다. ST_Within은 ST_Contains의 정확히 반대 결과를 리턴합니다.

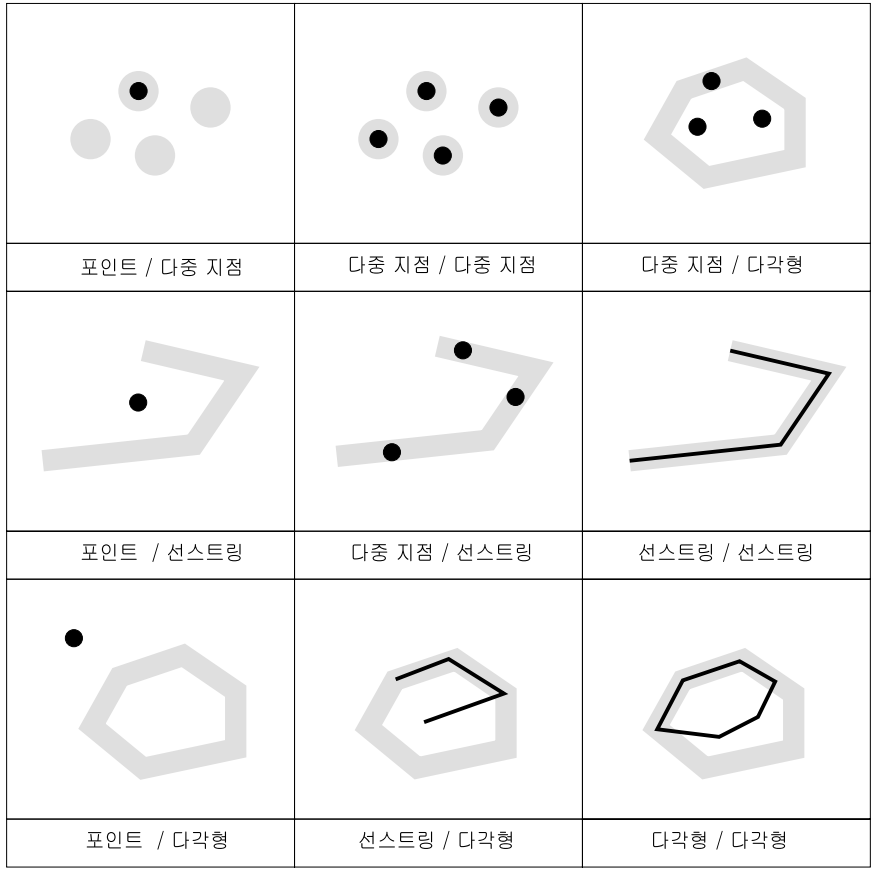


그림 18. Within

ST_Within 술어 패턴 행렬은 양쪽 기하학의 내부가 교차해야 하고 1차 기하학(기하학 *a*)의 경계가 보조(기하학 *b*)의 외부와 교차하지 말아야 함을 나타냅니다.

표 54. ST_Within에 대한 행렬

		b		
		내부	**경계**	**외부**
a	**내부**	T	*	F
	경계	*	*	F
	외부	*	*	*

자세한 내용은 297 페이지의 『ST_Within』를 참조하십시오.

ST_Contains

ST_Contains는 두 번째 기하학이 완전하게 첫번째 기하학에 포함되면 1(TRUE)을 리턴합니다. ST_Contains 술어는 ST_Within 술어의 정확히 반대 결과를 리턴합니다.

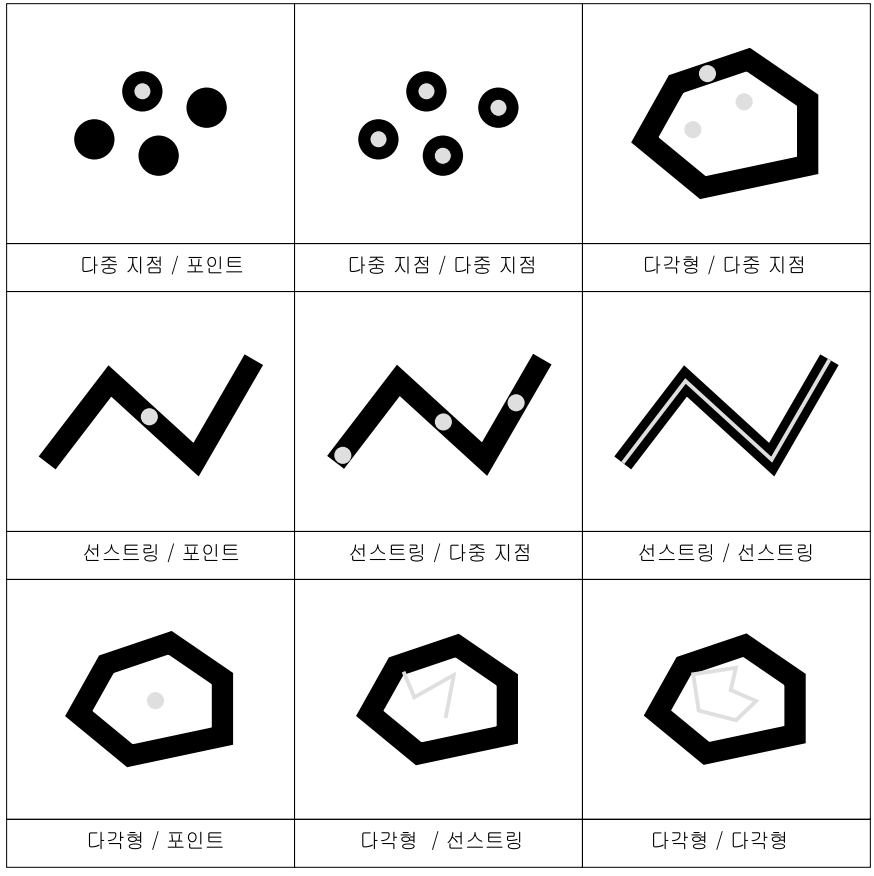


그림 19. ST_Contains

ST_Contains 술어 패턴 행렬은 양쪽 기하학의 내부가 교차해야 하고 보조 기하학(기하학 *b*)의 경계가 1차(기하학 *a*)의 외부와 교차하지 말아야 함을 나타냅니다.

표 55. ST_Contains에 대한 행렬

		b		
		내부	**경계**	**외부**
a	**내부**	T	*	*
	경계	*	*	*
	외부	F	F	*

자세한 내용은 208 페이지의 『ST_Contains』를 참조하십시오.

ST_Relate

ST_Relate 함수는 두 기하학을 비교하여 기하학들이 DE-9IM 패턴 행렬 문자열에 의해 지정된 조건을 충족하면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)이 리턴됩니다. 자세한 내용은 287 페이지의 『ST_Relate』를 참조하십시오.

ST_Distance

ST_Distance 함수는 두 구분 지형을 분리시키는 최소 거리를 보고합니다. 지형이 구분되지 않으면 함수가 0 최소 거리를 보고합니다.

예를 들어, ST_Distance는 비행기가 두 위치 사이에 비행할 수 있는 최단 거리를 보고할 수 있습니다. 그림20에서 이에 대해 설명합니다.

그림 20. 두 도시 사이의 최단 거리. ST_Distance는 LA와 시카고의 위치 좌표를 입력으로 취하고 이 위치들 사이의 최단 거리를 나타내는 값을 리턴할 수 있습니다.

자세한 내용은 221 페이지의 『ST_Distance』를 참조하십시오.

기존 기하학에서 새 기하학을 생성하는 함수

DB2 Spatial Extender는 기존 기하학에서 새 기하학을 생성하는 술어 및 변환 함수를 제공합니다.

ST_Intersection

ST_Intersection 함수는 두 기하학의 교차 세트를 리턴합니다. 교차 세트는 항상 소스 기하학의 최소 차원인 콜렉션으로 리턴됩니다. 예를 들어, 다각형을 교차하는 선스트링의 경우에 교차 함수는 다각형의 내부와 경계에 공통적인 선스트링의 해당 부분으로 구성된 다중 선스트링을 리턴합니다. 소스 선스트링이 둘 이상의 비

연속 세그먼트가 있는 다각형과 교차하는 경우에는 다중 선스트링에 둘 이상의 선스트링이 들어 있습니다. 기하학들이 교차하지 않거나 교차한 결과 차원이 두 소스 기하학 차원수 보다 적으면, 빈 기하학이 리턴됩니다.

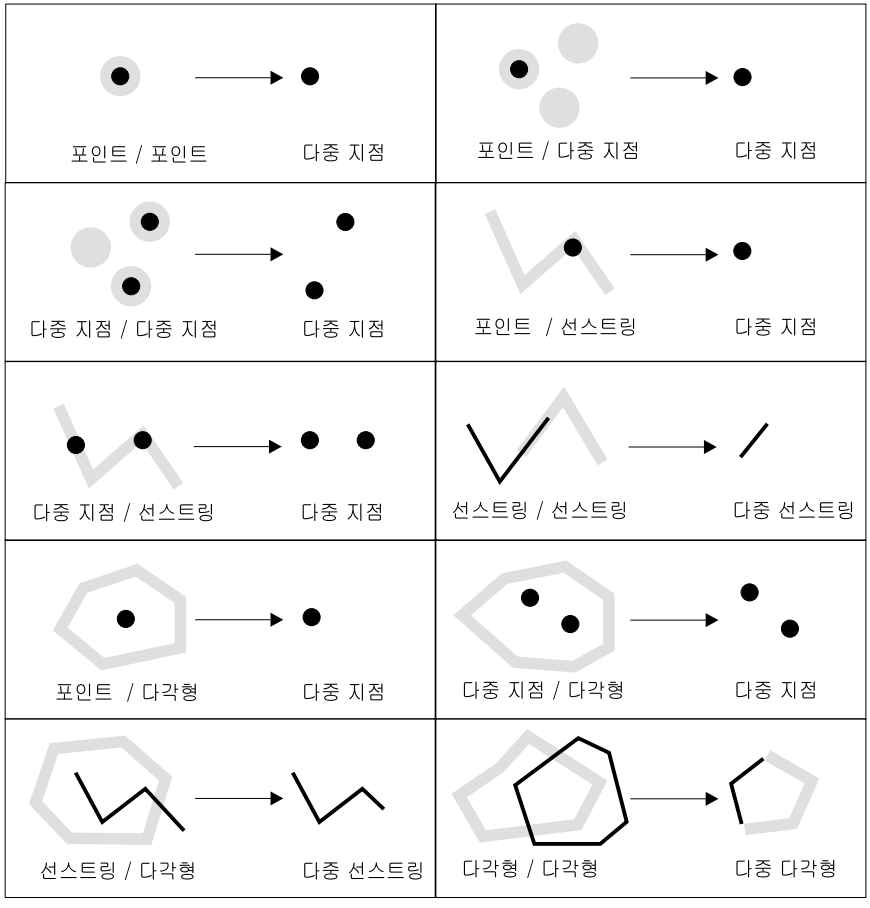


그림 21. ST_Intersection. ST_Intersection 함수 예.

자세한 내용은 241 페이지의 『ST_Intersection』를 참조하십시오.

ST_Difference

ST_Difference 함수는 보조 기하학에 의해 교차되지 않은 1차 기하학의 부분을 리턴합니다. 이는 공간의 논리적 AND NOT입니다. ST_Difference 함수는 차원

과 유사한 기하학에 대해서만 작용하며 소스 기하학과 같은 차원수를 가진 콜렉션을 리턴합니다. 소스 기하학이 동일한 이벤트에서는 빈 기하학이 리턴됩니다.

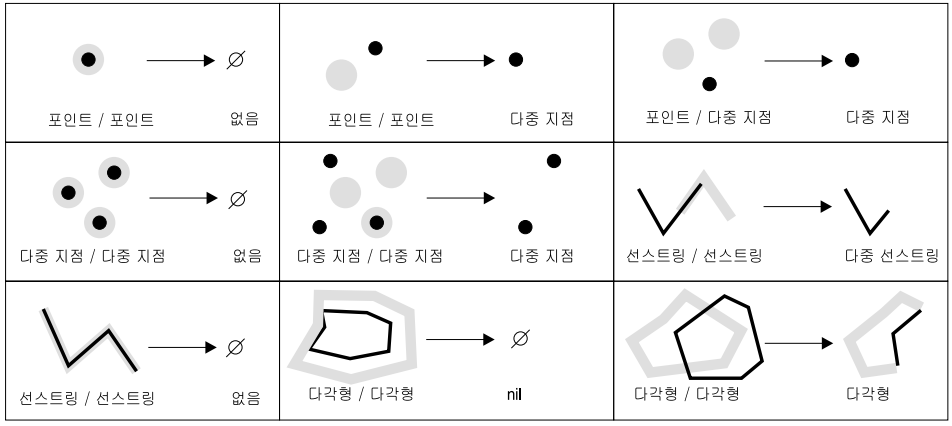


그림 22. ST_Difference

자세한 내용은 216 페이지의 『ST_Difference』를 참조하십시오.

ST_Union

ST_Union 함수는 두 기하학의 통합 세트를 리턴합니다. 이는 공간의 논리 OR입니다. 소스 기하학들은 차원이 유사해야 합니다. ST_Union은 항상 결과를 콜렉션으로 리턴합니다.

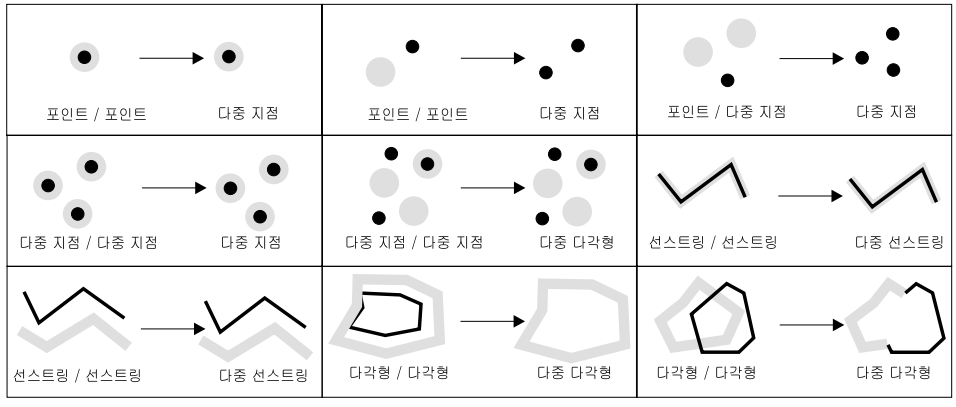


그림 23. ST_Union

자세한 내용은 296 페이지의 『ST_Union』를 참조하십시오.

ST_Buffer

ST_Buffer 함수는 지정된 간격에 있는 기하학을 둘러싼 기하학을 생성합니다. 다각형은 1차 기하학이 버퍼화될 때 또는 콜렉션 요소가 모든 버퍼 다각형이 겹칠 정도로 충분히 가까워질 때마다 생겨납니다. 그러나 버퍼화된 콜렉션의 요소 사이에 충분한 간격이 있을 때, ST_Buffer 함수가 다중 다각형을 리턴하는 경우에는 개별적인 버퍼 다각형이 생성됩니다.

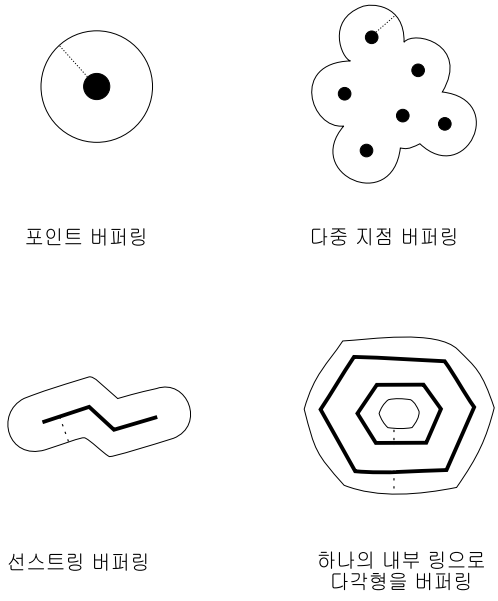


그림 24. ST_Buffer

ST_Buffer 함수는 양수 및 음수 간격 둘 다를 허용하지만, 2차원 기하학(다각형 및 다중 다각형)만이 음수 버퍼에 적용됩니다. 버퍼 간격의 절대값은 소스 기하학의 차원이 2보다 적을 때마다(다각형 또는 다중 다각형 이외의 모든 기하학) 사용됩니다.

일반적으로 외부 링의 경우에 양수 버퍼 간격은 소스 기하학의 중심에서 멀리 떨어진 다각형 링을 생성하고 음수 버퍼 간격은 중심을 향하고 있는 다각형 또는 다중 다각형 링을 생성합니다. 다각형 또는 다중 다각형의 내부 링 경우에, 양수 버퍼 간격은 중심을 향하고 있는 버퍼링을 생성하고 음수 버퍼 간격은 중심에서 멀리 떨어진 버퍼링을 생성합니다.

버퍼화 프로세스에서는 겹치는 다각형을 병합합니다. 다각형의 최대 내부 폭의 한 배 반보다 큰 음수 간격의 경우에는 결국 빈 기하학이 됩니다.

자세한 내용은 205 페이지의 『ST_Buffer』를 참조하십시오.

LocateAlong

치수가 있는 기하학의 경우에 특정 치수의 위치는 LocateAlong 함수로 찾을 수 있습니다. LocateAlong은 위치를 다중점으로 리턴합니다. 소스 기하학의 차원이 0이면(예: 점 및 다중점), 일치하는 정도가 정확히 일치해야 하고 일치하는 치수 값을 가진 그러한 점들이 다중점으로 리턴됩니다. 그러나 0보다 큰 차원의 소스 기하학 경우에는 위치가 써넣어집니다. 예를 들어, 입력된 치수 값이 5.5이고 선스트링의 정점상의 치수가 각각 3, 4, 5, 6 및 7이면, 치수 값 5와 6인 정점 사이에 정확히 반이 되는 써넣어진 점이 리턴됩니다.

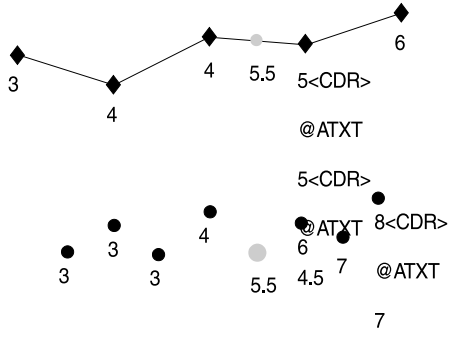


그림 25. LocateAlong

자세한 내용은 183 페이지의 『LocateAlong』를 참조하십시오.

LocateBetween

LocateBetween 함수는 치수가 있는 소스 기하학으로부터의 두 치수 값 사이에 놓여 있는 경로 또는 위치 세트 중 하나를 리턴합니다. 소스 기하학 차원이 0이면 LocateBetween은 치수가 두 소스 치수 사이에 놓여 있는 모든 점을 포함하는 다중점을 리턴합니다. 소스 기하학 차원이 0보다 큰 경우에, 경로를 써넣을 수 있으면 LocateBetween이 다중 선스트링을 리턴하고 그렇지 않으면 LocateBetween이 점 위치가 들어 있는 다중점을 리턴합니다. LocateBetween이 경로를 써넣을 수 없거나 치수들 사이의 위치를 찾을 수 없을 때마다 빈 점이 리턴됩니다.

LocateBetween은 기하학의 모든 것을 포함한 탐색을 수행합니다. 그러므로 기하학의 치수는 *from* 치수보다 크거나 같고 *to* 치수 보다 작거나 같아야 합니다.

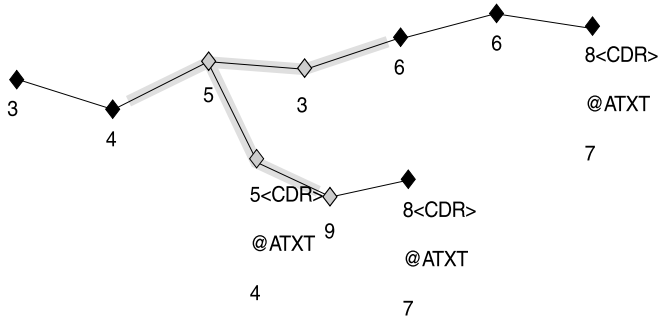


그림 26. LocateBetween

자세한 내용은 185 페이지의 『LocateBetween』를 참조하십시오.

ST_ConvexHull

ST_ConvexHull 함수는 볼록 형상을 형성하는 적어도 세 개의 정점이 있는 모든 다각형의 볼록 덮개 다각형을 리턴합니다. 기하학의 정점이 볼록 형상을 형성하지 않으면 ST_ConvexHull이 널을 리턴합니다. ST_ConvexHull은 자주 점 세트로부터 TIN 네트워크를 작성하는 데 사용되는 모자이크 세공의 첫 단계가 됩니다.

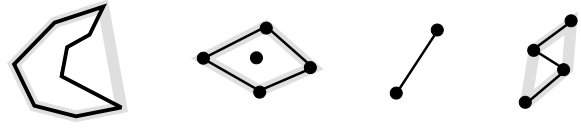


그림 27. ST_ConvexHull

자세한 내용은 210 페이지의 『ST_ConvexHull』를 참조하십시오.

ST_Polygon

선스트링에서 다각형을 생성합니다. 자세한 내용은 286 페이지의 『ST_Polygon』를 참조하십시오.

기하학값의 형식을 변환하는 함수

DB2 Spatial Extender는 세 가지 GIS 데이터 교환 형식을 지원합니다.

- 잘 알려진 텍스트 표현식
- 잘 알려진 2진 표현식
- ESRI 2진 형상 표현식

잘 알려진 텍스트 표현식

DB2 Spatial Extender에는 텍스트 설명으로부터 기하학을 생성하는 여러 함수가 있습니다.

ST_WKTToSQL

기하학 유형의 텍스트 표현식으로부터 기하학을 작성합니다. 어떠한 공간 참조 시스템 식별자도 지정하지 말아야 합니다. 자세한 내용은 301 페이지의 『ST_WKTToSQL』를 참조하십시오.

ST_GeomFromText

기하학 유형의 텍스트 표현식으로부터 기하학을 작성합니다. 공간 참조 시스템 식별자가 지정되어야 합니다. 자세한 내용은 229 페이지의 『ST_GeometryFromText』를 참조하십시오.

ST_PointFromText

점 텍스트 표현식으로부터 점을 작성합니다. 자세한 내용은 276 페이지의 『ST_PointFromText』를 참조하십시오.

ST_LineFromText

행스트링 텍스트 표현식으로부터 행 스트링을 작성합니다. 자세한 내용은 256 페이지의 『ST_LineFromText』를 참조하십시오.

ST_PolyFromText

다각형 텍스트 표현식으로부터 다각형을 작성합니다. 자세한 내용은 283 페이지의 『ST_PolyFromText』를 참조하십시오.

ST_MPointFromText

다중점 표현식으로부터 다중점을 작성합니다. 자세한 내용은 262 페이지의 『ST_MPointFromText』를 참조하십시오.

ST_MLineFromText

다중 행스트링 표현식으로부터 다중 행스트링을 작성합니다. 자세한 내용은 259 페이지의 『ST_MLineFromText』를 참조하십시오.

ST_MPolyFromText

다중 다각형 표현식으로부터 다중 다각형을 작성합니다. 자세한 내용은 265 페이지의 『ST_MPolyFromText』를 참조하십시오.

텍스트 표현식은 ASCII 문자열입니다. 이 표현식을 이용해 기하학을 ASCII 텍스트 양식으로 교환할 수 있습니다. 이 함수는 특수 프로그램 구조의 정의를 2진 표현으로 맵핑하지 않아도 됩니다. 그러므로 이 함수는 3GL 또는 4GL 프로그램에서 사용할 수 있습니다.

ST_AsText 함수는 기존 기하학값을 텍스트 표현식으로 변환합니다. 자세한 내용은 202 페이지의 『ST_AsText』를 참조하십시오.

317 페이지의 『OGC 잘 알려진 텍스트 표현식』에서 잘 알려진 텍스트 표현식에 대한 자세한 내용을 참조하십시오.

잘 알려진 2진 표현식

DB2 Spatial Extender에는 잘 알려진 2진(WKB) 표현식으로부터 기하학을 생성하는 여러 함수가 있습니다.

ST_WKBToSQL

기하학 유형의 잘 알려진 2진 표현식으로부터 기하학을 작성합니다. 어떠한 공간 참조 시스템 식별자도 지정하지 말아야 합니다. 자세한 내용은 299 페이지의 『ST_WKBToSQL』를 참조하십시오.

ST_GeomFromWKB

기하학 유형의 잘 알려진 2진 표현식으로부터 기하학을 작성합니다. 공간 참조 시스템 식별자가 지정되어야 합니다. 자세한 내용은 231 페이지의 『ST_GeomFromWKB』를 참조하십시오.

ST_PointFromWKB

점의 잘 알려진 2진 표현식으로부터 점을 작성합니다. 자세한 내용은 277 페이지의 『ST_PointFromWKB』를 참조하십시오.

ST_LineFromWKB

선스트링의 잘 알려진 2진 표현식으로부터 선스트링을 작성합니다. 자세한 내용은 257 페이지의 『ST_LineFromWKB』를 참조하십시오.

ST_PolyFromWKB

다각형의 잘 알려진 2진 표현식으로부터 다각형을 작성합니다. 자세한 내용은 284 페이지의 『ST_PolyFromWKB』를 참조하십시오.

ST_MPointFromWKB

다중점의 잘 알려진 2진 표현식으로부터 다중점을 작성합니다. 자세한 내용은 263 페이지의 『ST_MPointFromWKB』를 참조하십시오.

ST_MLineFromWKB

다중 선스트링의 잘 알려진 2진 표현식으로부터 다중 선스트링을 작성합니다. 자세한 내용은 260 페이지의 『ST_MLineFromWKB』를 참조하십시오.

ST_MPolyFromWKB

다중 다각형의 잘 알려진 2진 표현식으로부터 다중 다각형을 작성합니다. 자세한 내용은 266 페이지의 『ST_MPolyFromWKB』를 참조하십시오.

잘 알려진 2진 표현식은 연속적인 바이트 스트림입니다. ODBC 클라이언트와 SQL 데이터베이스 사이에 기하학을 2진 양식으로 교환할 수 있습니다. 이 기하학 함수에는 2진 표현식을 맵핑하는 데 C 구조의 정의가 필요합니다. 그래서 이는 3GL 프로그램 내에서 사용될 예정이며 4GL 환경에는 적합하지 않습니다.

ST_AsBinary 함수는 기존 기하학값을 잘 알려진 2진 표현식으로 변환합니다. 자세한 내용은 201 페이지의 『ST_AsBinary』를 참조하십시오.

322 페이지의 『OGC 잘 알려진 2진(WKB) 표현식』에서 잘 알려진 2진 표현식에 대한 자세한 내용을 참조하십시오.

ESRI 형상 표현식

DB2 Spatial Extender에는 ESRI 형상 표현식으로부터 기하학을 생성하는 여러 함수가 있습니다. ESRI 형상 표현식은 텍스트와 잘 알려진 2진 표현식에 의해 지원되는 2차원 표현식 이외에도 Z 좌표와 치수를 지원합니다.

ShapeToSQL

기하학 유형의 형상으로부터 기하학을 작성합니다. 어떠한 공간 참조 시스템 식별자도 지정하지 말아야 합니다. 자세한 내용은 197 페이지의 『ShapeToSQL』를 참조하십시오.

GeometryFromShape

기하학 유형의 형상으로부터 기하학을 작성합니다. 공간 참조 시스템 식별자가 지정되어야 합니다. 자세한 내용은 175 페이지의 『GeometryFromShape』를 참조하십시오.

PointFromShape

점 형상으로부터 점을 작성합니다. 자세한 내용은 194 페이지의 『PointFromShape』를 참조하십시오.

LineFromShape

polyline 형상으로부터 행스트링을 작성합니다. 자세한 내용은 181 페이지의 『LineFromShape』를 참조하십시오.

PolyFromShape

polyline 형상으로부터 다각형을 작성합니다. 자세한 내용은 195 페이지의 『PolyFromShape』를 참조하십시오.

MPointFromShape

다중점 형상으로부터 다중점을 작성합니다. 자세한 내용은 190 페이지의 『MPointFromShape』를 참조하십시오.

MLineFromShape

다중 파트 polyline 형상으로부터 다중 선스트링을 작성합니다. 자세한 내용은 188 페이지의 『MLine FromShape』를 참조하십시오.

MPolyFromShape

다중 파트 다각형 형상으로부터 다중 다각형을 작성합니다. 자세한 내용은 192 페이지의 『MPolyFromShape』를 참조하십시오.

이 함수의 일반적인 구문은 동일합니다. 첫번째 인수는 BLOB 데이터 유형으로 입력된 형상 표현식입니다. 두 번째 인수는 기하학에 지정될 공간 참조 식별자입니다. 예를 들어, GeometryFromShape 함수의 구문은 다음과 같습니다.

```
GeometryFromShape(shapegeometry, SRID)
```

2진 표현식을 맵핑하려면 이 형상 함수에 C 구조의 정의가 필요합니다. 그래서 이는 3GL 프로그램 내에서 사용될 예정이며 4GL 환경에는 적합하지 않습니다.

AsBinaryShape 함수는 기하학 값을 ESRI 형상 표현식으로 변환합니다. 자세한 내용은 174 페이지의 『AsBinaryShape』를 참조하십시오.

327 페이지의 『ESRI 형상 표현식』에서 형상 표현식에 대한 자세한 내용을 참조하십시오.

제14장 SQL 조회에 대한 공간 함수

이 장에서는 공간 데이터 조회시 사용 가능한 함수들의 리스트를 보여줍니다. 각각의 함수에 대해서는 섹션별로 구문, 리턴 유형 및 코드 예가 설명되어 있습니다. 이 장의 일부 예에서는 하나 이상의 컬럼이 공간 컬럼으로 정의되는 CREATE TABLE문을 포함하고 있습니다.

다음의 고려사항이 공간 함수에 적용됩니다.

- 이 장의 예는 라이브러리 이름 db2gse로 규정됩니다. 각각의 공간 함수 및 유형을 db2gse로 명시적 규정을 하는 대신에, db2gse가 포함되도록 함수 경로를 설정할 수 있습니다.
- 공간 컬럼에 데이터를 삽입하기 전에 다음을 수행하십시오.
  - udf_mem_sz 매개변수의 *크기*를 증가시키십시오. 제안하는 초기 설정값은 2048 입니다. 2048 값이 적절하지 않으면, 256 단위로 udf_mem_sz 매개변수의 *크기*를 증가시키십시오.
  - 공간 컬럼을 계층으로 등록하십시오. 공간 컬럼 등록에 대한 자세한 정보는 39 페이지의 『제4장 공간 컬럼 정의, 공간 컬럼을 계층으로 등록 및 지오코더를 사용하여 공간 컬럼을 유지보수』를 참조하십시오.

AsBinaryShape

AsBinaryShape은 기하학 오브젝트를 취하고, BLOB을 리턴합니다.

구문

db2gse.AsBinaryShape(g db2gse.ST_Geometry)

리턴 유형

BLOB(1m)

예

다음의 코드 조각은 AsBinaryShape 함수가 SENSITIVE_AREAS 테이블의 존 다각형을 형상 다각형으로 변환하는 방법을 예시합니다. 이 형상 다각형은 응용프로그램의 화면표시용 draw_polygon 함수에 전달됩니다.

```
/* SQL 표현식을 작성합니다. */
strcpy(sqlstmt, "select db2gse.AsBinaryShape (zone) from SENSITIVE_AREAS
where db2gse.EnvelopesIntersect(zone, db2gse.PolyFromShape(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문을 준비합니다. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* 형상의 pcbvalue1 길이를 설정합니다. */
pcbvalue1 = blob_len;

/* 형상 매개변수를 바인드합니다. */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB, blob_len,
0, shape, blob_len, &pcbvalue1);

/* 조회를 실행합니다. */
rc = SQLExecute (hstmt);

/* 조회 (존 다각형)의 결과를
   사전 추출된 2진 변수에 지정합니다. */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* 표시 창에서 각각의 다각형을 사전 추출하고 이를 표시합니다. */

while(SQL_SUCCESS == (rc = SQLFetch(hstmt)))
  draw_polygon(fetched_binary);
```

GeometryFromShape

GeometryFromShape은 형상 및 공간 참조 시스템 아이덴티티를 취해 기하학 오브젝트를 리턴합니다.

구문

db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)

리턴 유형

db2gse.ST_Geometry

예

다음의 C 코드 조각에는 LOTS 테이블에 데이터를 삽입하는 DB2 Spatial Extender SQL 함수와 함께 삽입된 ODBC 함수가 들어 있습니다.

LOTS 테이블은 각 구획을 고유하게 나타내는 LOT_ID 컬럼과 각 구획의 기하학이 들어 있는 LOT 다각형 컬럼으로 작성됩니다.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot     db2gse.ST_MultiPolygon);
```

GeometryFromShape 함수는 형상을 DB2 Spatial Extender 기하학으로 변환합니다. 전체 INSERT문이 shp_sql로 복사됩니다. INSERT문은 LOT_ID 데이터 및 LOT 데이터를 동적으로 수용할 수 있는 매개변수 표시문자를 포함합니다.

```
/* lot ID 및 lot 다중 다각형을 상주시키기 위한
   SQL INSERT문을 작성합니다. 물음표는 매개변수 표시문자이며,
   이는 런타임에서 검색되는 lot_id 및 lot 값을
   지시합니다. */
strcpy (shp_sql,"insert into LOTS (lot_id, lot) values (?, db2gse.GeometryFromShape
(cast(? as blob(1m)), db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* 정수 키 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* 형상을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

EnvelopesIntersect

EnvelopesIntersect는 두 기하학의 가장자리가 교차되면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.EnvelopesIntersect(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

get_window 함수는 응용프로그램으로부터 화면표시 창의 좌표를 검색합니다. 창 매개변수는 화면표시 다각형을 나타내는 좌표 문자열이 들어 있는 실제 다각형 형상 구조입니다. PolyFromShape 함수는 화면표시 창 형상을 EnvelopesIntersect 함수가 이의 교차 가장자리로 사용하는 DB2 Spatial Extender 다각형으로 변환합니다. 화면표시 창의 내부 또는 경계와 교차하는 모든 SENSITIVE_AREAS 존 다각형들이 리턴됩니다. 각 다각형은 결과 세트로부터 사전 추출되어 draw_polygon 함수로 전달됩니다.

```
/* 다각형 형상으로 표시 창 좌표를 가져옵니다.
get_window(&window)

/* SQL 표현식을 작성합니다. db2gse.EnvelopesIntersect 함수를
  사용하여 표시 창의 엔벨로프(envelope)를 교차하는
  결과 세트를 해당 존 다각형으로만 제한합니다. */
strcpy(sqlstmt, "select db2gse.AsBinaryShape(zone) from SENSITIVE_AREAS where
db2gse.EnvelopesIntersect (zone, db2gse.PolyFromShape(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* blob_len을 5 점 형상 다각형의 바이트 길이로
설정합니다. */
blob_len = 128;

/* SQL문을 준비합니다. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* pcbvalue1을 창 형상으로 설정합니다. */
pcbvalue1 = blob_len;

/* 형상 매개변수를 바인드합니다. */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB, blob_len,
0, window, blob_len, &pcbvalue1);

/* 조회를 실행합니다. */
rc = SQLExecute (hstmt);
```

```
/* 조회 결과, (존 다각형)을
   사전 추출된_2진 변수에 지정합니다. */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* 표시 창에서 각각의 다각형을 사전 추출하고 이를 표시합니다. */
while(SQL_SUCCESS == (rc = SQLFetch(hstmt))
  draw_polygon(fetched_binary);
```

Is3d

Is3d는 기하학 오브젝트를 취해 오브젝트에 3D 좌표가 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.Is3d(g db2gse.ST_Geometry)

리턴 유형

정수

예

다음의 CREATE TABLE문은 정수 유형의 GID 컬럼과 G1 기하학 컬럼 두 개가 있는 THREED_TEST문을 작성합니다.

```
CREATE TABLE THREED_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

INSERT문은 THREED_TEST 테이블에 두 점을 삽입합니다. 첫번째 점에는 Z 좌표가 들어 있지 않으나 두 번째에는 들어 있습니다.

```
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)))
```

다음의 SELECT문은 Is3d 함수의 결과와 함께 GID 컬럼의 내용을 나열합니다. 함수는 Z 좌표가 없는 첫번째 행에 대해서는 0을 리턴하고 Z 좌표가 있는 두 번째 행에 대해서는 1을 리턴합니다.

```
SELECT gid, db2gse.Is3d (g1) "Is it 3d?" FROM THREED_TEST
```

다음의 결과 세트가 리턴됩니다.

```
gid      Is it 3d?
------   ----------
     1            0
     2            1
```

IsMeasured

IsMeasured는 기하학 오브젝트를 취해 오브젝트를 측정하면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.IsMeasured(g db2gse.ST_Geometry)

리턴 유형

정수

예

다음 CREATE TABLE문은 행을 고유하게 나타내는 GID 컬럼과 점 기하학을 저장하는 G1 컬럼이 있는 MEASURE_TEST 테이블을 작성합니다.

```
CREATE TABLE MEASURE_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

다음의 INSERT문은 MEASURE_TEST 테이블에 두 개의 레코드를 삽입합니다. 첫번째 레코드는 치수가 없는 점을 저장합니다. 두 번째 레코드의 점에는 치수가 있습니다.

```
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)))
```

다음의 SELECT문과 해당 결과 세트에서는 IsMeasured 함수의 결과와 함께 GID 컬럼의 내용을 보여 줍니다. 첫번째 행에는 점에 치수가 없기 때문에 IsMeasured 함수가 0을 리턴합니다. 두 번째 행에는 점에 치수가 있기 때문에 1을 리턴합니다.

```
SELECT gid, db2gse.IsMeasured (g1) "Has measures?" FROM MEASURE_TEST
gid      Has measures
------   ----------
     1            0
     2            1
```

LineFromShape

LineFromShape는 유형 점 형상과 공간 참조 시스템 아이덴티티를 취해 선스트링을 리턴합니다.

구문

db2gse.Line FromShape(ShapeLineString Blob(1M), cr db2gse.coordref)

리턴 유형

db2gse.ST_LineString

예

다음의 코드 조각은 각 송수관선의 고유 ID, 크기 클래스 및 기하학을 가진 SEWERLINES 테이블을 상주시킵니다.

CREATE TABLE문은 세 개의 행이 있는 SEWERLINES 테이블을 작성합니다. 첫번째 컬럼인 SEWER_ID는 각 송수관선을 고유하게 식별합니다. 정수 유형인 두 번째 컬럼 CLASS는 송수관선의 유형을 나타내며 이는 일반적으로 행의 용량과 연관됩니다. 선스트링 유형인 세 번째 컬럼 SEWER는 송수관선의 기하학을 저장합니다.

```
CREATE TABLE SEWERLINES (sewer_id   integer, class  integer,
                         sewer db2gse.ST_LineString);

/* sewer_id, 크기 class 및 sewer 선스트링을 상주시키기
   위한 SQL INSERT문을 작성합니다. 물음표는 매개변수 표시문자이며,
   이는 런타임에 검색되는 sewer_id, class 및 sewer 기하학 값을
   지시합니다. */
strcpy (shp_sql,"insert into sewerlines (sewer_id,class,sewer)
values (?,?, db2gse.Line FromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* 정수 키 값을 첫번째 매개변수에 바인드합니다. */
```

```
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_id, 0, &pcbvalue1);

/* 정수 class 값을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = 0;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_class, 0, &pcbvalue2);;

/* 형상을 세 번째 매개변수에 바인드합니다. */
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, sewer_shape, blob_len, &pcbvalue3);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

LocateAlong

LocateAlong은 기하학 오브젝트 및 치수를 취하여 측정시 발견된 점 세트를 다중점으로서 리턴합니다.

구문

db2gse.LocateAlong(g db2gse.ST_Geometry, adistance Double)

리턴 유형

db2gse.ST_Geometry

예

다음의 CREATE TABLE문은 LOCATEALONG_TEST 테이블을 작성합니다. LOCATEALONG_TEST에는 각 행을 고유하게 나타내는 GID 컬럼과 샘플 기하학을 저장하는 G1 기하학 컬럼이 있습니다.

```
CREATE TABLE LOCATEALONG_TEST (gid integer, g1 db2gse.ST_Geometry)
```

다음의 INSERT문은 두 행을 삽입합니다. 첫번째는 다중 선스트링이고 두 번째는 다중점입니다.

```
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)))
```

다음의 SELECT문과 해당 결과 세트에서 LocateAlong 함수는 치수가 6.5인 점을 찾도록 지정됩니다. 첫번째 행에서는 두 점이 들어 있는 다중점을 리턴합니다. 그러나 두 번째 행에서는 빈 점을 리턴합니다. 선형 기능(0보다 큰 차원의 기하학)의 경우에는 LocateAlong이 점을 써넣지만 다중점의 경우에는 목표 측정이 정확하게 일치해야 합니다.

```
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.
```

다음의 SELECT문과 해당 결과 세트에서 LocateAlong 함수는 양쪽 행에 대해 다중 점을 리턴합니다. 목표 치수 7은 다중 선스트링 및 다중 점 소스 데이터 모두에서 치수와 일치합니다.

```
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

LocateBetween은 기하학 오브젝트와 두 개의 측정 위치를 취하고, 두 측정 위치 사이에 연결이 단절된 경로의 세트를 나타내는 기하학을 리턴합니다.

구문

db2gse.LocateBetween(g db2gse.ST_Geometry, adistance Double, anotherdistance Double)

리턴 유형

db2gse.ST_Geometry

예

다음의 CREATE TABLE문은 LOCATEBETWEEN_TEST 테이블을 작성합니다. LOCATEBETWEEN_TEST에는 각 행을 고유하게 나타내는 GID 컬럼과 샘플 기하학을 저장하는 G1 다중 선스트링 컬럼이 있습니다.

```
CREATE TABLE LOCATEBETWEEN_TEST (gid integer, g1 db2gse.ST_Geometry)
```

다음의 INSERT문은 LOCATEBETWEEN_TEST 테이블에 두 개의 행을 삽입합니다. 첫번째 행은 다중 선스트링이고 두 번째는 다중점입니다.

```
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)))
```

다음의 SELECT문과 해당 결과 세트에서는 LocateBetween 함수가 치수 6.5와 7.5 포함 사이에 위치하는 치수를 찾는 방법을 보여 줍니다. 첫번째 행에서는 여러 선스트링이 들어 있는 다중 선스트링을 리턴합니다. 두 번째 행에서는 소스 데이터가 다중점이기 때문에 다중점을 리턴합니다. 소스 데이터의 차원이 0(점 또는 다중점)일 때는 완전하게 일치해야 합니다.

```
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

M은 점을 취해 이의 치수를 리턴합니다.

구문

db2gse.M(p db2gse.ST_Point)

리턴 유형

Double

예

다음의 CREATE TABLE문은 M_TEST 테이블을 작성합니다. M_TEST에는 행을 고유하게 나타내는 GID 정수 컬럼과 샘플 기하학을 저장하는 PT1 점 컬럼이 있습니다.

```
CREATE TABLE M_TEST (gid integer, pt1 db2gse.ST_Point)
```

다음의 INSERT문은 치수를 가진 점이 들어 있는 행과 치수가 없는 점이 들어 있는 행을 삽입합니다.

```
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)))
```

다음의 SELECT문과 해당 결과 세트에서 M 함수는 점의 측정값을 나열합니다. 첫번째 점에는 치수가 없기 때문에 M 함수가 NULL을 리턴합니다.

```
SELECT gid, db2gse.M (pt1) "The measure" FROM M_TEST

GID         The measure
----------- ------------------------
          1                         -
          2   +7.00000000000000E+000

  2 record(s) selected.
```

MLine FromShape

MLineFromShape는 유형 다중 선스트링 형상과 공간 참조 시스템 아이덴티티를 취해 다중 선스트링을 리턴합니다.

구문

db2gse.MLineFromShape(ShapeMultiLineString Blob(1M), cr db2gse.coordref)

리턴 유형

db2gse.ST_MultiLineString

예

다음의 코드 조각은 고유 ID, 이름 및 수역 다중 선스트링을 가진 WATERWAYS 테이블을 데이터 상주시킵니다.

WATERWAYS 테이블은 테이블에 저장된 각 하천과 강 체계를 나타내는 ID 및 NAME 컬럼으로 작성됩니다. 강 및 하천 체계는 자주 여러 선스트링의 총계가 되기 때문에 WATER 컬럼은 다중 선스트링입니다.

```
CREATE TABLE WATERWAYS (id        integer,
                        name      varchar(128),
                        water     db2gse.ST_MultiLineString);

/* ID, name 및 다중 선스트링을 상주시키기 위한
   SQL문을 작성합니다. 물음표는 매개변수 표시문자이며, 이는
   런타임에 검색되는 ID, name 및 water 값을
   지시합니다. */
strcpy (shp_sql,"insert into WATERWAYS (id,name,water)
values (?,?, db2gse.MLineFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* 정수 id 값을 첫번째 매개변수에 바인드합니다. */

pcbvalue1 = 0;
```

```
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);
/* varchar name 값을 두 번째 매개변수에 바인드합니다. */

pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, name_len, 0, &name, name_len, &pcbvalue2);

/* 형상을 세 번째 매개변수에 바인드합니다. */
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, water_shape, blob_len, &pcbvalue3);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

MPointFromShape

MPointFromShape는 유형 다중점 형상과 공간 참조 시스템 아이덴티티를 취해 다중점을 리턴합니다.

구문

db2gse.MPointFromShape(ShapeMultiPoint (1M), srs db2gse.coordref)

리턴 유형

db2gse.ST_MultiPoint

예

이 코드 조각은 생물학자의 SPECIES_SITINGS 테이블을 상주시킵니다.

SPECIES_SITINGS 테이블은 세 개의 컬럼으로 작성됩니다. SPECIES 및 GENUS 컬럼은 각 행을 고유하게 나타내며 SITINGS 다중점은 종 구획(species sitings)의 위치를 저장합니다.

```
CREATE TABLE SPECIES_SITINGS (species  varchar(32),
                              genus varchar(32),
                              sitings  db2gse.ST_MultiPoint);

/* species, genus 및 sitings를 상주시키기 위한 SQL INSERT문을
   작성합니다. 물음표는 매개변수 표시문자이며, 이는
   런타임에 검색되는 name 및 water 값을 지시합니다. */
strcpy (shp_sql,"insert into SPECIES_SITINGS (species,genus,sitings)
values (?,?, db2gse.MPointFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* varchar species 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = species_len;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, species_len, 0, species, species_len, &pcbvalue1);

/* varchar genus 값을 두 번째 매개변수에 바인드합니다. */
```

```
pcbvalue2 = genus_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, genus_len, 0, name, genus_len, &pcbvalue2);

/* 형상을 세 번째 매개변수에 바인드합니다. */
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, sitings_len, 0, sitings_shape, sitings_len, &pcbvalue3);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

MPolyFromShape

MPolyFromShape는 유형 다중 다각형 형상과 공간 참조 시스템 아이덴티티를 취해 다중 다각형을 리턴합니다.

구문

db2gse.MPolyFromShape(ShapeMultiPolygon Blob(1m), srs db2gse.coordref)

리턴 유형

db2gse.ST_MultiPolygon

예

이 코드 조각에서는 LOTS 테이블을 데이터 상주시킵니다.

LOTS 테이블은 각 구획을 고유하게 나타내는 LOT_ID와 구획선 기하학이 포함된 LOT 다중 다각형을 저장합니다.

```
CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );

/* lot_id 및 lot를 상주시키기 위한 SQL INSERT문을 작성합니다.
   물음표는 매개변수 표시문자이며, 이는 런타임에 검색되는 lot_id 및
   값을 지시합니다. */
strcpy (shp_sql,"insert into LOTS (lot_id,lot)
values (?, db2gse.MPolyFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* lot_id 정수 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* lot 형상을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = lot_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, lot_len, 0, lot_shape, lot_len, &pcbvalue2);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

PointFromShape

PointFromShape는 유형 점 형상과 공간 참조 시스템 아이덴티티를 취해 점을 리턴합니다.

구문

db2gse.PointFromShape(db2gse.ShapePoint blob(1M), srs db2gse.coordref)

리턴 유형

db2gse.ST_Point

예

이 프로그램 조각에서는 HAZARDOUS_SITES 테이블을 상주시킵니다.

모험적인 사이트는 다음에 나오는 CREATE TABLE문으로 작성되는 HAZARDOUS_SITES 테이블에 저장됩니다. 점으로 정의되는 위치 컬럼은 각 모험 사이트의 지리적 중심지인 위치를 저장합니다.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);

/* site_id, name 및 location을 상주시키기 위한 SQL INSERT문을
   작성합니다. 물음표는 매개변수 표시문자이며, 이는
   런타임에 검색되는 site_id, name 및 location 값을 지시합니다. */
strcpy (shp_sql,"insert into HAZARDOUS_SITES (site_id, name, location)
values (?,?, db2gse.PointFromShape (cast(? as blob(1m)), db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* site_id 정수 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);

/* name varchar 값을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* location 형상을 세 번째 매개변수에 바인드합니다. */
pcbvalue3 = location_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, location_len, 0, location_shape, location_len, &pcbvalue3);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

PolyFromShape

PolyFromShape는 유형 다각형 형상과 공간 참조 시스템 아이덴티티를 취해 다각형을 리턴합니다.

구문

db2gse.PolyFromShape (ShapePolygon Blob(1M), srs db2gse.coordref)

리턴 유형

db2gse.ST_Polygon

예

프로그램 조각에서는 SENSITIVE_AREAS 테이블을 상주시킵니다. 물음표는 런타임시 검색될 id, name, size, type 및 zone 값들에 대한 매개변수 표시문자를 나타냅니다.

SENSITIVE_AREAS 테이블에는 공공빌딩의 다각형 기하학을 저장하는 존 컬럼 외에도 위협적인 공공빌딩을 서술하는 여러 컬럼들이 들어 있습니다.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);

/* name, size, type 및 zone을 상주시키기 위한 SQL INSERT문을
   작성합니다. 물음표는 매개변수 표시문자이며, 이는
   런타임에 검색되는 id, name, size, type 및 zone 값을 지시합니다. */
strcpy (shp_sql,"insert into SENSITIVE_AREAS (id, name, size, type, zone)
values (?,?,?,?, db2gse.PolyFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* id 정수 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);
/* name varchar 값을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);
```

```
/* size float를 세 번째 매개변수에 바인드합니다. */
pcbvalue3 = 0;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_FLOAT,
     SQL_REAL, 0, 0, &size, 0, &pcbvalue3);

/* type varchar을 네 번째 매개변수에 바인드합니다. */
pcbvalue4 = type_len;
rc = SQLBindParameter (hstmt, 4, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_VARCHAR, type_len, 0, type, type_len, &pcbvalue4);

/* zone 다각형을 다섯 번째 매개변수에 바인드합니다. */
pcbvalue5 = zone_len;
rc = SQLBindParameter (hstmt, 5, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, zone_len, 0, zone_shp, zone_len, &pcbvalue5);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

ShapeToSQL

ShapeToSQL은 잘 알려진 2진 표현식으로 주어진 db2gse.ST_Geometry 값을 구성합니다. SRID 값으로 0이 자동으로 사용됩니다.

구문

db2gse.ST_ShapeToSQL(ShapeGeometry blob(1M))

리턴 유형

db2gse.ST_Geometry

예

다음의 C 코드 조각에는 LOTS 테이블에 데이터를 삽입하는 DB2 Spatial Extender SQL 함수와 함께 삽입된 ODBC 함수가 들어 있습니다. LOTS 테이블은 각 구획을 고유하게 식별하는 lot_id 컬럼, 그리고 각 구획의 기하학이 들어 있는 구획 다각형 컬럼의 두 개로 작성되었습니다.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

ShapeToSQL 함수는 형상을 DB2 Spatial Extender 기하학으로 변환합니다. 전체 INSERT문이 shp_sql로 복사됩니다. INSERT문에는 LOT_id 및 구획 데이터를 동적으로 허용하는 매개변수 표시문자가 들어 있습니다.

```
/* lot ID 및 lot 다중 다각형을 상주시키기 위한
   SQL INSERT문을 작성합니다. 물음표는 매개변수 표시문자이며,
   이는 런타임에서 검색되는 lot_id 및 lot 값을
   지시합니다. */

strcpy (shp_sql,"insert into lots (lot_id, lot)
values(?, db2gse.ShapeToSQL(cast(? as blob(1m))))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */

rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */

rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);
```

```
/* 정수 키 값을 첫번째 매개변수에 바인드합니다. */

pcbvalue1 = 0;

rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* 형상을 두 번째 매개변수에 바인드합니다. */

pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* INSERT문을 실행합니다. */

rc = SQLExecute (hstmt);
```

ST_Area

ST_Area는 다각형 또는 다중 다각형을 취해 이의 영역을 리턴합니다.

구문

db2gse.ST_Area(s db2gse.ST_Surface)

리턴 유형

Double

예

도시 공학자는 빌딩 영역 목록이 필요합니다. 목록을 얻기 위해 GIS 기술자는 빌딩 ID와 각 빌딩의 위치 영역을 선택합니다.

빌딩 위치는 다음의 CREATE TABLE문으로 작성되는 BUILDINGFOOTPRINTS 테이블에 저장됩니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

도시 공학자의 요구를 충족시키기 위해 기술자는 다음의 SELECT문을 사용하여 BUILDINGFOOTPRINTS 테이블에서 고유키, 빌딩_ID 및 각 빌딩 위치의 영역을 선택합니다.

```
SELECT building_id, db2gse.ST_Area (footprint) "Area"
FROM BUILDINGFOOTPRINTS;
```

SELECT문은 다음의 결과 세트를 리턴합니다.

```
building_id   Area
------------  ------------------------
        506    +1.40768000000000E+003
       1208    +2.55759000000000E+003
        543    +1.80786000000000E+003
        178    +2.08671000000000E+003
        .
        .
        .
```

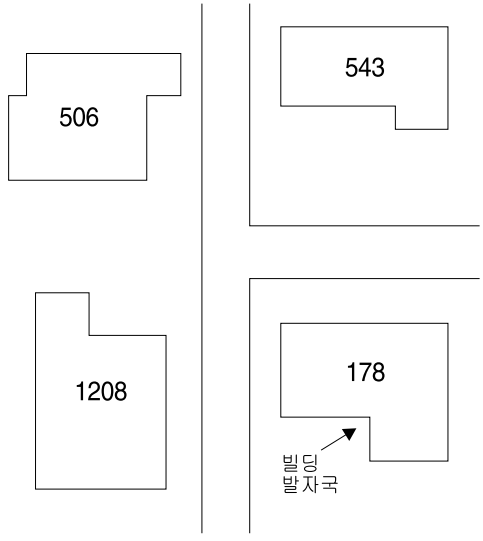


그림 28. 빌딩 위치를 찾는 데 영역 사용. 빌딩 ID 번호가 붙은 네 개의 빌딩 위치가 이의 인접한 거리와 함께 표시됩니다.

ST_AsBinary

ST_AsBinary는 기하학 오브젝트를 취해 이의 잘 알려진 2진 표현식을 리턴합니다.

구문

db2gse.ST_AsBinary(g db2gse.ST_Geometry)

리턴 유형

BLOB(1m)

예

다음 코드 조각에서는 ST_AsBinary 함수가 BUILDINGFOOTPRINTS 테이블의 위치 다중 다각형을 WKB 다중 다각형으로 변환하는 방법에 대해 설명합니다. 이 다중 다각형은 응용프로그램의 화면표시용 draw_polygon 함수에 전달됩니다.

```
/* SQL 표현식을 작성합니다. */
strcpy(sqlstmt, "select db2gse.ST_AsBinary (footprint) from BUILDINGFOOTPRINTS
where db2gse.EnvelopesIntersect(footprint, db2gse.ST_PolyFromWKB(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문을 준비합니다. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* 형상의 pcbvalue1 길이를 설정합니다. */
pcbvalue1 = blob_len;

/* 형상 매개변수를 바인드합니다. */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB, blob_len,
0, shape, blob_len, &pcbvalue1);

/* 조회를 실행합니다. */
rc = SQLExecute (hstmt);

/* 조회 (존 다각형)의 결과를
   사전 추출된_2진 변수에 지정합니다.
 */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* 표시 창에서 각각의 다각형을 사전 추출하고 이를 표시합니다. */
while(SQL_SUCCESS == (rc = SQLFetch(hstmt)))
  draw_polygon(fetched_binary);
```

ST_AsText

db2gse.ST_AsText는 기하학 오브젝트를 취해 이의 잘 알려진 텍스트 표현식을 리턴합니다.

구문

db2gse.ST_AsText(g db2gse.ST_Geometry)

리턴 유형

Varchar(4000)

예

다음 시나리오에서는 db2gse.ST_AsText 함수가 HAZARDOUS_SITES 위치 점을 이의 텍스트 설명으로 변환합니다.

```
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;
```

SELECT문은 다음의 결과 세트를 리턴합니다.

```
SITE_ID Name                                Location
------- ----------------------------------- ----------------------------------------
    102 W. H. Kleenare Chemical Repository  POINT (1020.00000000 324.00000000)
```

ST_Boundary

ST_Boundary는 기하학 오브젝트를 취해 이의 결합된 경계를 기하학 오브젝트로서 리턴합니다.

구문

db2gse.ST_Boundary(g db2gse.ST_Geometry)

리턴 유형

db2gse.ST_Geometry

예

다음의 코드 조각에서는 BOUNDARY_TEST라는 테이블이 작성됩니다. BOUNDARY_TEST에는 varchar로 정의되는 GEOTYPE 컬럼과 슈퍼클래스 기하학으로 정의되는 G1이 있습니다. 다음에 나오는 INSERT문은 각 하나씩의 서브클래스 기하학을 삽입합니다. ST_Boundary 함수는 G1 기하학 컬럼에 저장된 각 서브클래스의 경계를 검색합니다. 결과 기하학의 차원은 항상 입력 기하학 보다 하나 적다는 점에 유의하십시오. 점 및 다중 점의 결과는 항상 빈 기하학, 차원 1인 경계입니다. 선스트링 및 다중 선스트링은 다중점 경계를 리턴합니다. 다각형 또는 다중 다각형은 항상 다중 선스트링 경계, 차원 1을 리턴합니다.

```
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('MultipolygonT',
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_Buffer는 기하학 오브젝트와 거리를 취해 소스 오브젝트를 둘러싸고 있는 기하학 오브젝트를 리턴합니다.

구문

db2gse.ST_Buffer(g db2gse.ST_Geometry , adistance Double)

리턴 유형

db2gse.ST_Geometry

예

지역 감독자에게는 반경 5마일 내에 학교, 병원 및 사설 요양원과 같은 주요 지역이 겹지는 위험 사이트 목록이 필요합니다. 중요 영역은 다음의 CREATE TABLE 문으로 작성된 테이블 SENSITIVE_AREAS에 저장됩니다. ZONE 컬럼은 다각형으로 정의되며 이 컬럼에는 각 주요 지역의 윤곽이 저장됩니다.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

위험 사이트는 다음의 CREATE TABLE문으로 작성되는 HAZARDOUS_SITES 테이블에 저장됩니다. 점으로 정의된 LOCATION 컬럼은 각 위험 사이트의 지리적 중심이 되는 위치를 저장합니다.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

SENSITIVE_AREAS 및 HAZARDOUS_SITES 테이블은 db2gse.ST_Overlaps 함수에 의해 조인됩니다. 함수는 존 다각형이 버퍼화된 반경 5마일의 HAZARDOUS_SITES 위치 점과 겹치는 모든 SENSITIVE_AREAS 행에 대해 1(TRUE)을 리턴합니다.

```
SELECT sa.name "Sensitive Areas", hs.name "Hazardous Sites"
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Overlaps(sa.zone, db2gse.ST_Buffer (hs.location,(5 * 5280))) = 1;
```

그림29에서는 이 관리 구역에 있는 일부 주요 지역들이 위험 사이트 위치의 5마일 버퍼 내에 있습니다. 양쪽 5마일 버퍼들이 병원에서 교차되며 버퍼들 중 하나는 학교에서 교차됩니다. 그러나 사설 요양원은 양쪽 반경의 외부에 안전하게 놓여 있습니다.

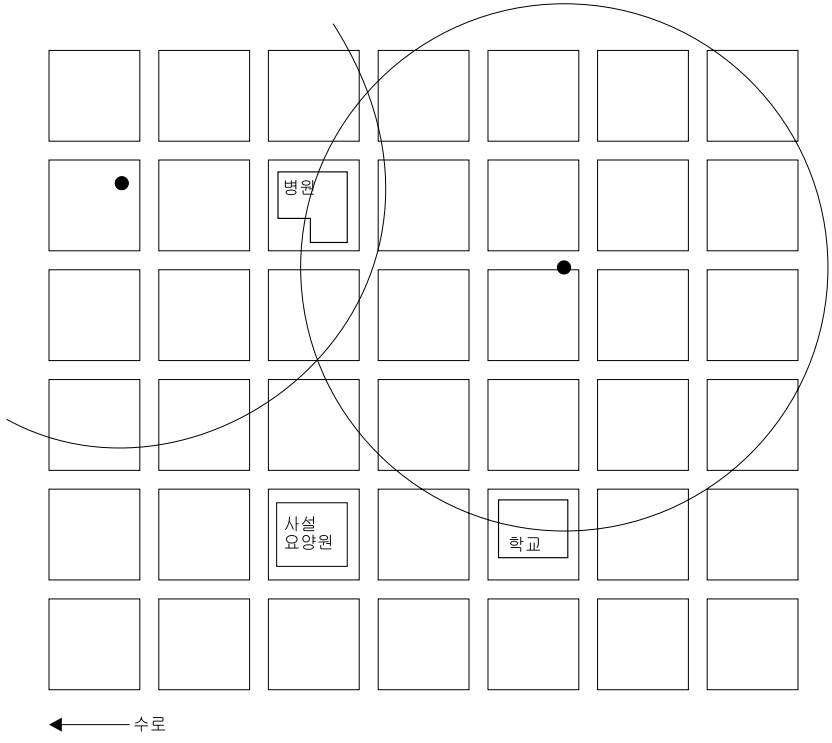


그림 29. 5마일 반경의 버퍼가 점에 적용됨

ST_Centroid

ST_Centroid는 다각형 또는 다중 다각형 또는 다중 다각형을 취해 이의 지리적 중심을 점으로서 리턴합니다.

구문

db2gse.ST_Centroid(s db2gse.ST_Surface) db2gse.ST_Centroid(ms db2gse.ST_MultiSurface)

리턴 유형

For surface: db2gse.ST_Point

예

도시 GIS 기술자는 빌딩 위치의 다중 다각형을 빌딩 밀도 그래픽에 단일 점으로서 표시하려고 합니다.

빌딩 위치는 다음의 CREATE TABLE문으로 작성되는 BUILDINGFOOTPRINTS 테이블에 저장됩니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
              lot_id      integer,
              footprint   db2gse.ST_MultiPolygon);
```

ST_Centroid 함수는 각 빌딩 위치 다중 다각형의 중심을 리턴합니다. AsBinaryShape 함수는 중심 점을 응용프로그램에서 인식하는 외부적 표현인 형상으로 변환합니다.

```
SELECT building_id,
       CAST(db2gse.AsBinaryShape(db2gse.ST_Centroid (footprint))
                    as blob(1m)) "Centroid"
FROM BUILDINGFOOTPRINTS;
```

ST_Contains

ST_Contains는 두 개의 기하학 오브젝트를 취해 첫번째 오브젝트에 두 번째 오브젝트가 완벽하게 포함되면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_Contains(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

아래의 예에서는 두 개의 테이블이 작성됩니다. 한 테이블에는 도시의 빌딩 위치가 들어 있고 다른 테이블에는 이의 구획이 들어 있습니다. 도시 공학자는 모든 빌딩 위치가 이의 구획 안에 완벽하게 있는지 확인하려 합니다.

양 테이블에서 다중 다각형 데이터 유형은 빌딩 위치와 구획의 기하학을 저장합니다. 데이터베이스 설계자는 양쪽 기능 모두에 대해 다중 다각형을 선택했습니다. 설계자는 구획이 강과 같은 자연적인 기능에 의해 결합이 해제될 수 있고 빌딩 위치가 자주 여러 빌딩들로 구성될 수 있음을 알고 있습니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                          lot_id      integer,
                          footprint   db2gse.ST_MultiPolygon);
CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

도시 공학자는 먼저 한 구획 내에 완전히 포함되지 않은 빌딩들을 선택합니다.

```
SELECT building_id
  FROM BUILDINGFOOTPRINTS, LOTS
WHERE db2gse.ST_Contains(lot,footprint) = 0;
```

도시 공학자는 첫번째 조회에서 구획 다각형의 외부에 위치하는 모든 빌딩의 목록을 리턴한다는 것을 알게 됩니다. 그러나 도시 공학자는 이 정보가 다른 빌딩들이 빌딩에 지정된 올바른 구획 ID를 가지고 있는지 여부를 지정하지는 않는다는 것도 압니다. 이 두 번째 조회에서는 BUILDINGFOOTPRINTS 테이블의 lot_id 컬럼에 대해 데이터 무결성 점검을 수행합니다.

```
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;
```

그림30에서 빌딩 ID가 붙은 빌딩 위치가 이의 구획 내에 놓여 있습니다. 구획선은 점선으로 설명됩니다. 표시되지 않더라도, 이 선들은 거리 중앙선까지 확장되며 구획과 그 구획 내의 빌딩 위치를 완벽하게 에워 쌉니다.

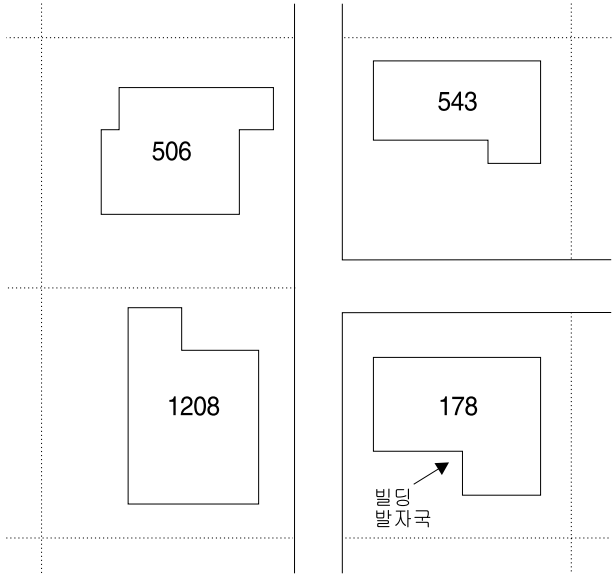


그림 30. ST_Contains를 사용하여 구획 내에 모든 빌딩이 포함되는지 확인

ST_ConvexHull

ST_ConvexHull는 기하학 오브젝트를 취해 볼록 덮개를 리턴합니다.

구문

db2gse.ST_ConvexHull(g db2gse.ST_Geometry)

리턴 유형

db2gse.ST_Geometry

예

이 예에서는 GEOTYPE 및 G1 두 개의 컬럼이 있는 CONVEXHULL_TEST 테이블을 작성합니다. varchar(20)인 GEOTYPE 컬럼은 기하학으로 정의된 G1에 저장된 기하학의 서브클래스 이름을 저장합니다.

```
CREATE TABLE CONVEXHULL_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

각 INSERT문은 CONVEXHULL_TEST 테이블에 각 서브클래스 유형의 기하학을 삽입합니다.

```
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)))
```

다음의 SELECT문은 GEOTYPE 컬럼 및 convex hull에 저장된 서브클래스 이름을 나열합니다. ST_ConvexHull 함수에 의해 생성된 convexhull은 ST_AsText 함수에 의해 텍스트로 변환됩니다. 그런 다음, 이는 ST_AsText의 기본 출력이 varchar(4000)이기 때문에 varchar(256)으로 배정됩니다.

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText(db2gse.ST_ConvexHull(G1)))
            as varchar(256) "The convexhull"
FROM CONVEXHULL_TEST
```

ST_CoordDim

ST_CoordDim은 ST_Geometry 값의 좌표 차원을 리턴합니다. 좌표 차원에 대한 설명은 141 페이지의 『점』을 참조하십시오.

구문

db2gse.ST_CoordDim(g1 db2gse.ST_Geometry)

리턴 유형

정수

예

coorddim_test 테이블은 컬럼 geotype 및 g1으로 작성됩니다. geotype 컬럼은 g1 기하학 컬럼에 저장된 기하학 서브클래스의 이름을 저장합니다.

```
CREATE TABLE coorddim_test (geotype varchar(20), g1 db2gse.ST_Geometry)
```

INSERT문은 coorddim_test 테이블에 샘플 서브클래스를 삽입합니다.

```
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))
)
```

SELECT문은 해당 geotype의 좌표 차원으로 geotype 컬럼에 저장된 서브클래스 이름을 나열합니다.

```
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_Crosses는 두 개의 기하학 오브젝트를 취해 이의 교차 결과가 소스 오브젝트의 최대 차원 보다 하나 작은 차원을 가진 기하학 오브젝트인 경우에는 1(TRUE)을 리턴합니다. 교차 오브젝트에는 양쪽 소스 기하학에 대해 내부인 점이 들어 있으며 소스 오브젝트 중 하나와 동일하지 않습니다. 그렇지 않은 경우에는 0(FALSE)을 리턴합니다.

구문

db2gse.ST_Crosses(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

도청에서는 관내에 모든 위험 폐기물 저장 설비가 수로의 5마일 이내에 위치할 수 없다고 명시한 새로운 규정을 고려중입니다. 도 GIS 관리 프로그램은 강과 하천의 정확한 표시를 가지고 있으며 이는 WATERWAYS 테이블에 다중 선스트링으로 저장됩니다. 그러나, GIS 관리 프로그램에서는 각 위험 폐기물 저장 설비에 대해 단일 점 위치만을 가집니다.

```
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);
```

도 감독자는 제안된 규정을 위반하는 설비에 경고해야 할 필요가 있는지 판단하기 위해, GIS 관리 프로그램은 위험 사이트 위치를 버퍼화하여 강이나 하천 버퍼 다각형을 통과하는지 알아야 합니다. ST_Crosses 술어는 버퍼화된 HAZARDOUS_SITES와 WATERWAYS를 비교합니다. 그래서, 도에서 제안한 규정 반지름을 수로가 통과하는 그러한 레코드만이 리턴됩니다.

```
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;
```

그림31에서 위험 폐기물 사이트의 5마일 버퍼가 도의 관리 지구를 관통하여 흐르는 하천 네트워크와 엇갈립니다. 하천 네트워크는 다중 선스트링으로 정의된 상태입니다. 그러므로, 결과 세트에는 반지름을 가로지르는 그러한 세그먼트들의 일부분인 모든 선스트링 세그먼트가 포함됩니다.

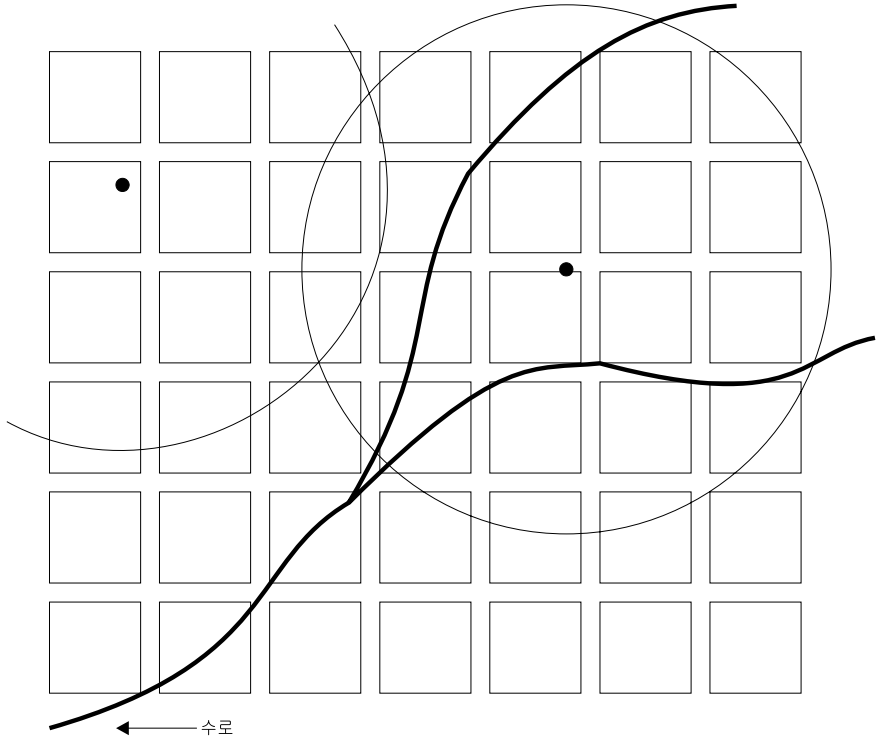


그림 31. ST_Crosses를 사용하여 위험 폐기물 영역을 통과하는 수로 찾기

ST_Difference

ST_Difference는 두 개의 기하학 오브젝트를 취하여 소스 오브젝트와 차이가 있는 기하학 오브젝트를 리턴합니다.

구문

db2gse.ST_Difference(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

db2gse.ST_Geometry

예

도시 공학자는 빌딩에서 다루지 않는 도시 구획 영역의 총 역역을 알고 있어야 합니다. 즉, 도시 공학자는 빌딩 영역을 제거한 이후의 구획 영역 합을 원합니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                              footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS ( lot_id   integer,
                    lot     db2gse.ST_MultiPolygon);
```

도시 공학자는 lot_id에 대한 BUILDINGFOOTPRINTS 및 LOTS 테이블을 결합합니다. 그런 다음, 공학자는 구획에서 빌딩 위치를 뺀 차이 영역의 합을 취합니다.

```
SELECT SUM(db2gse.ST_Area(db2gse.ST_Difference(lot,footprint)))
  FROM BUILDINGFOOTPRINTS bf, LOTS
WHERE bf.lot_id = LOTS.lot_id;
```

ST_Dimension

ST_Dimension은 기하학 오브젝트를 취해 이의 차원을 리턴합니다.

구문

db2gse.ST_Dimension(g1 db2gse.ST_Geometry)

리턴 유형

정수

예

DIMENSION_TEST 테이블은 컬럼 GEOTYPE 및 G1으로 작성됩니다. GEOTYPE 컬럼은 G1 기하학 컬럼에 저장된 기하학 서브클래스의 이름을 저장합니다.

```
CREATE TABLE DIMENSION_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

INSERT문은 DIMENSION_TEST 테이블에 샘플 서브클래스를 삽입합니다.

```
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)))
```

다음의 SELECT문은 해당 geotype의 차원으로 GEOTYPE 컬럼에 저장된 서브 클래스 이름을 나열합니다.

```
SELECT geotype, db2gse.ST_Dimension(g1) "The dimension"
FROM DIMENSION_TEST
```

다음의 결과 세트가 리턴됩니다.

```
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_Disjoint는 두 개의 기하학을 취해, 두개의 기하학이 교차한 결과 빈 세트가 생성되면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_Disjoint(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

보험 회사에서는 마을의 병원, 사설 요양원 및 학교에 대한 보험 적용범위를 평가해야 합니다. 이러한 과정의 일부분으로 위험 폐기물 사이트가 각 공공 빌딩에 미치는 영향을 판단하는 작업도 포함됩니다. 보험 회사에서는 오염될 우려가 없는 그러한 공공 빌딩만 고려하고자 합니다. 보험 회사에서 고용한 GIS 자문위원은 위험 폐기물 사이트의 5마일 반경 내에 있지 않는 모든 공공 빌딩을 찾도록 의뢰받았습니다.

SENSITIVE_AREAS 테이블에는 공공빌딩의 다각형 기하학을 저장하는 존 컬럼 외에도 위협적인 공공빌딩을 서술하는 여러 컬럼들이 들어 있습니다.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

HAZARDOUS_SITES 테이블은 SITE_ID 및 NAME 컬럼에 사이트의 아이덴티티를 저장하는 반면에, 각 사이트의 실제 지리적 위치는 LOCATION 컬럼에 저장됩니다.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

다음의 SELECT문에서는 위험 폐기물 사이트의 5마일 반지름 이내에 없는 모든 중요 영역의 이름을 나열합니다. ST_Intersects 함수는 함수 결과가 1 대신 0으로

설정되면 이 조회에 ST_Disjoint 함수를 대체할 수 있습니다. 이는 ST_Intersects와 ST_Disjoint가 정확히 반대 결과를 리턴하기 때문입니다.

```
SELECT sa.name
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Disjoint(db2gse.ST_Buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

그림32에서 중요 사이트는 위험 폐기물 사이트의 5마일 반지름과 비교됩니다. ST_Disjoint 함수가 1(TRUE)을 리턴하는 유일한 중요 영역은 사설 요양소입니다. ST_Disjoint 함수는 두 개의 기하학이 어떤 방법으로도 교차하지 않을 때마다 1을 리턴합니다.

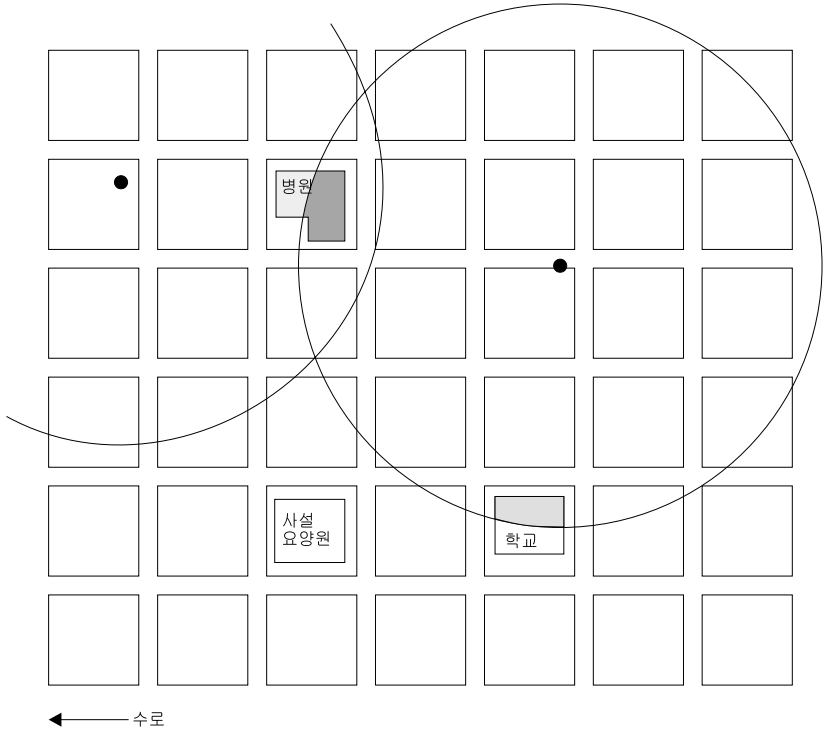


그림 32. ST_Disjoint를 사용하여 위험 폐기물 영역 내에 놓여 있지(교차하지) 않은 빌딩을 찾으십시오.

ST_Distance

ST_Distance는 두 개의 기하학을 취해 그들 사이의 가장 먼 거리를 리턴합니다.

구문

db2gse.ST_Distance(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

Double

예

도시 공학자는 구획선의 1 피트 내에 있는 모든 빌딩 목록이 필요합니다.

BUILDINGFOOTPRINTS 테이블의 BUILDING_ID 컬럼은 고유하게 각 빌딩을 식별합니다. LOT_ID 컬럼은 각 빌딩이 속해 있는 구획을 나타냅니다. 위치 다중 다각형은 각 빌딩 위치의 기하학을 저장합니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                            footprint   db2gse.ST_MultiPolygon);
```

LOTS 테이블은 각 구획을 고유하게 나타내는 구획 ID와 구획 선 기하학이 포함된 구획 다중 다각형을 저장합니다.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot     db2gse.ST_MultiPolygon);
```

조회에서는 구획선의 1 피트 내에 있는 빌딩 ID의 목록을 리턴합니다. ST_Distance 함수는 위치와 구획 다중 다각형의 경계 사이에 공간 조인을 수행합니다. 그러나, bf.lot_id와 LOTS.lot_id 사이의 equijoin은 동일한 구획에 속해 있는 다중 다각형만이 ST_Distance 함수에 의해 비교되도록 합니다.

```
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_Endpoint는 선스트링을 취해 선스트링의 최종 점인 점 하나를 리턴합니다.

구문

db2gse.ST_Endpoint(c db2gse.ST_Curve)

리턴 유형

db2gse.ST_Point

예

ENDPOINT_TEST 테이블은 선스트링을 저장하는 각 행과 LN1 선스트링 컬럼을 고유하게 나타내는 GID 정수 컬럼을 저장합니다.

```
CREATE TABLE ENDPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

INSERT문은 ENDPOINT_TEST 테이블에 선스트링을 삽입합니다. 첫번째 명령문에는 Z 좌표나 치수가 들어 있지 않으며 두 번째에는 들어 있습니다.

```
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)))
```

다음의 SELECT문은 ST_Endpoint 함수의 결과와 함께 GID 컬럼을 나열합니다. ST_Endpoint 함수는 ST_AsText 함수에 의해 텍스트로 변환된 점 기하학을 생성합니다. CAST 함수는 기본 ST_AsText 함수의 기본 varchar(4000) 값을 varchar(60)으로 단축하는 데 사용됩니다.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.ST_Endpoint(ln1))
                          AS varchar(60)) "Endpoint"
FROM ENDPOINT_TEST
```

다음의 결과 세트가 리턴됩니다.

```
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_Envelope는 기하학 오브젝트를 취해 이를 에워싼 상자를 기하학으로서 리턴합니다.

구문

db2gse.ST_Envelope(g db2gse.ST_Geometry)

리턴 유형

db2gse.ST_Geometry

예

ENVELOPE_TEST 테이블의 GEOTYPE 컬럼은 G1 기하학 컬럼에 저장된 기하학 서브클래스의 이름을 저장합니다.

```
CREATE TABLE ENVELOPE_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

다음의 INSERT문은 ENVELOPE_TEST 테이블에 각 기하학 서브클래스를 삽입합니다.

```
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)))
```

다음의 SELECT문은 외피 옆에 있는 서브클래스 이름을 나열합니다. ST_Envelope 함수는 점, 선스트링 또는 다각형을 리턴하기 때문에, 이의 출력은 ST_AsText 함수에 의해 텍스트로 변환됩니다. CAST 함수는 ST_AsText 함수의 기본 varchar(4000) 결과를 varchar(280)으로 변환합니다.

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText
(db2gse.ST_Envelope(g1)) AS varchar(280)) "The envelope"
FROM ENVELOPE_TEST
```

다음의 결과 세트가 리턴됩니다.

```
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_Equals는 두 개의 기하학을 비교하여 기하학이 동일하면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_Equals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

도시 GIS 전문가는 BUILDINGFOOTPRINTS 테이블의 일부 데이터가 어느 정도 중복되어 있음을 예상하고 있습니다. 전문가는 테이블을 조회하여 위치의 다중 다각형이 동일한지 판단합니다.

BUILDINGFOOTPRINTS 테이블은 다음의 명령문으로 작성됩니다. BUILDING_ID 컬럼은 고유하게 빌딩을 나타내고, LOT_ID 컬럼은 빌딩의 구획을 나타내며, FOOTPRINT 컬럼은 빌딩의 기하학을 저장합니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                            footprint   db2gse.ST_MultiPolygon);
```

BUILDINGFOOTPRINTS 테이블은 ST_Equals 술어에 의해 그 자신에 공간적으로 결합되며 이 술어는 동일한 두 개의 다중 다각형을 찾을 때마다 1을 리턴합니다. bf1.building_id <> bf2.building_id 조건을 설정하면 동일한 기하학은 비교하지 않습니다.

```
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_ExteriorRing은 다각형을 취해 이의 외부 링을 선스트링으로 리턴합니다.

구문

db2gse.ST_ExteriorRing(s db2gse.ST_Polygon)

리턴 유형

db2gse.ST_LineString

예

여러 남해 섬에 사는 조류 집단을 연구하는 조류학자는 특정 조류의 생태 구역이 해안선에 제한됨을 알고 있습니다. 섬의 수용 용량을 계산하기 위해 조류학자는 섬의 주계(perimeter)가 필요합니다. 일부 섬에는 여러 개의 호수가 있다해도 호수의 해안선은 보다 공격적인 다른 조류 종이 독점적으로 서식하고 있습니다. 그러므로, 조류학자에게는 섬의 외부 링에 대한 주계가 필요합니다.

ISLANDS 테이블의 ID 및 NAME 컬럼은 각 섬을 나타내며 유형 ST_Polygon인 LAND 컬럼은 각각의 기하학을 저장합니다.

```
CREATE TABLE ISLANDS (id    integer,
                      name  varchar(32),
                      land  db2gse.ST_Polygon);
```

ST_ExteriorRing 함수는 각 섬 다각형에서 외부 링을 선스트링으로 추출합니다. 선스트링의 길이는 길이 함수에 의해 설정됩니다. 선스트링 길이는 SUM 함수로 요약됩니다.

```
SELECT SUM(db2gse.ST_length(db2gse.ST_ExteriorRing (land))) FROM ISLANDS;
```

228 페이지의 그림33에서 섬의 외부 링은 각각의 섬이 바다와 공유하는 생태학적 인터페이스를 나타냅니다. 일부 섬에는 다각형의 내부 링으로 표현되는 호수가 있습니다.

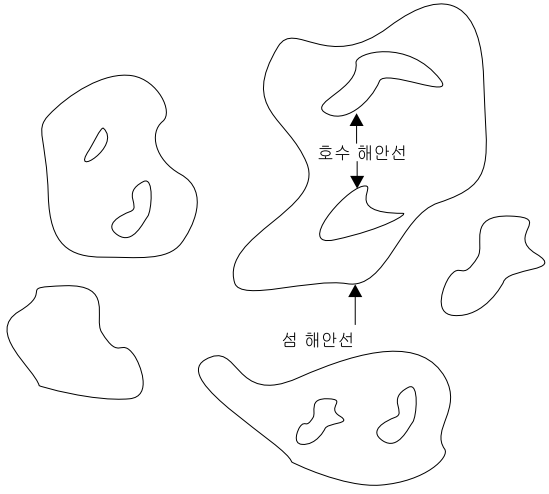


그림 33. ST_ExteriorRing을 사용하여 섬 해안선의 길이를 판별

ST_GeometryFromText

ST_GeometryFromText는 잘 알려진 텍스트 표현식 및 공간 참조 시스템 아이덴티티를 취해 기하학 오브젝트를 리턴합니다.

구문

db2gse.ST_GeometryFromText(geometryTaggedText Varchar(4000),cr db2gse.coordref)

리턴 유형

db2gse.ST_Geometry

예

GEOMETRY_TEST 테이블에는 각 행을 고유하게 나타내는 정수 GID 컬럼과 기하학을 저장하는 G1 컬럼이 들어 있습니다.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

INSERT문은 GEOMETRY_TEST 테이블의 GID 및 G1 컬럼에 데이터를 삽입합니다. ST_GeometryFromText 함수는 각 기하학의 텍스트 표현식을 이의 해당 DB2 Spatial Extender 인스턴스화 가능 서브클래스로 변환합니다.

```
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_GeomFromWKB는 잘 알려진 2진 표현식 및 공간 참조 시스템 아이덴티티를 취해 기하학 오브젝트를 리턴합니다.

구문

db2gse.ST_GeomFromWKB(WKBGeometry Blob(1M), cr db2gse.coordref)

리턴 유형

db2gse.ST_Geometry

예

다음의 C 코드 조각에는 LOTS 테이블에 데이터를 삽입하는 DB2 Spatial Extender SQL 함수와 함께 삽입된 ODBC 함수가 들어 있습니다.

LOTS 테이블은 각 구획을 고유하게 나타내는 LOT_ID 컬럼 및 각 구획의 기하학이 들어 있는 LOT 다각형 컬럼으로 작성됩니다.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot     db2gse.ST_MultiPolygon);
```

ST_GeomFromWKB 함수는 WKB 표현식을 DB2 Spatial Extender 기하학으로 변환합니다. 전체 INSERT문은 wkb_sql char 문자열로 복사됩니다. INSERT문은 LOT_ID 데이터 및 LOT 데이터를 동적으로 수용할 수 있는 매개변수 표시문자를 포함합니다.

```
/* lot ID 및 lot 다중 다각형을 상주시키기 위한
   SQL INSERT문을 작성합니다. 물음표는 매개변수 표시문자이며,
   이는 런타임에서 검색되는 lot_id 및 lot 값을
   지시합니다. */
strcpy (wkb_sql,"insert into LOTS (lot_id, lot) values (?, db2gse.ST_GeomFromWKB

(cast(? as blob(1m)), db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* 정수 키 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);
```

```
/* 형상을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

ST_GeometryN

ST_GeometryN은 콜렉션 및 정수 색인을 취해 콜렉션에 있는 *n* 번째 기하학 오브젝트를 리턴합니다.

구문

db2gse.ST GeometryN(g db2gse.ST_GeomCollection, n Integer)

리턴 유형

db2gse.ST_Geometry

예

도시 공학자는 빌딩 위치가 구획 다중 다각형의 첫번째 다각형 안에 모두 있는지 알아야 합니다.

BUILDING_ID 컬럼은 BUILDINGFOOTPRINTS 테이블의 각 행을 고유하게 식별합니다. LOT_ID 컬럼은 빌딩 구획을 나타냅니다. FOOTPRINT 컬럼은 빌딩의 기하학을 저장합니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                               footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS (lot_id   integer,
                   lot     db2gse.ST_MultiPolygon);
```

조회에서는 모두 첫번째 구획 다각형 내에 있는 모든 빌딩 위치의 BUILDINGFOOTPRINTS building_id 및 lot_id를 나열합니다. ST_GeometryN 함수는 다중 다각형 배열에서의 첫번째 구획 다각형을 리턴합니다.

```
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_GeometryType은 ST_Geometry 오브젝트를 취해 이의 기하학 유형을 문자열로 리턴합니다.

구문

db2gse.ST_GeometryType (g db2gse.ST_Geometry)

리턴 유형

Varchar(4000)

예

GEOMETRYTYPE_TEST 테이블에는 G1 기하학 컬럼이 들어 있습니다.

```
CREATE TABLE GEOMETRYTYPE_TEST(g1 db2gse.ST_Geometry)
```

다음의 INSERT문은 G1 컬럼에 각 기하학 서브클래스를 삽입합니다.

```
                          _TEST
VALUES(db2gse.ST_GeometryFromText('point (10.02 20.01)', db2gse.coordref()..srid(0)))

                          _TEST
VALUES (db2gse.ST_GeometryFromText('linestring (10.01 20.01, 10.01 30.01, 10.01 40.01)',
                           db2gse.coordref()..srid(0)))

                          _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))))

                          _TEST
VALUES(db2gse.ST_GeometryFromText('multipoint (10.02 20.01,10.32 23.98,11.92 25.64)',
                          db2gse.coordref()..srid(0)))

                          _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)))

                          _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)))
```

다음의 SELECT문은 G1 기하학 컬럼에 저장된 각 서브클래스의 기하학 유형을 나열합니다.

```
SELECT db2gse.ST_GeometryType(g1) "Geometry type" FROM GEOMETRYTYPE_TEST
```

다음의 결과 세트가 리턴됩니다.

```
Geometry type
--------------------------------
ST_Point
ST_LineString
ST_Polygon
ST_MultiPoint
ST_MultiLineString
ST_MultiPolygon

6 record(s) selected.
```

ST_InteriorRingN

다각형의 *n*번째 내부 링을 선스트링으로서 리턴합니다. 링은 기하학 방향에 의해 구성되지 않습니다. 링은 내부 기하학 검증 루틴에 의해 정의된 규칙에 따라 구성됩니다. 그러므로 링의 순서는 사전 정의할 수 없습니다.

구문

ST_InteriorRingN(p ST_Polygon, n Integer)

리턴 유형

db2gse.ST_LineString

예

여러 남해 섬에 사는 조류 집단을 연구하는 조류학자는 수동적인 특정 조류종의 생태 구역이 해안선에 제한되어 있음을 알고 있습니다. 일부 섬에는 여러 개의 호수가 있습니다. 호수의 해안선은 보다 공격적인 다른 조류 종이 독점적으로 서식하고 있습니다. 생태학자는 각 섬의 경우에 호수의 주계가 특정 임계값을 초과하면 공격적인 조류종이 너무 많아져서 수동적인 섬 해안선 조류종을 위협할 것임을 알고 있습니다. 그러므로, 조류학자에게는 섬의 내부 링에 대한 총 주계(perimeter)가 필요합니다.

237 페이지의 그림34에서 섬의 외부 링은 각각의 섬이 바다와 공유하는 생태학적 인터페이스를 나타냅니다. 일부 섬에는 다각형의 내부 링으로 표현되는 호수가 있습니다.

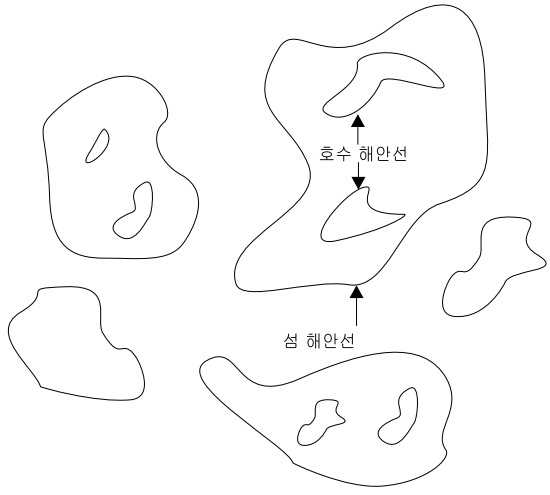


그림 34. ST_InteriorRingN을 사용하여 각 섬에 있는 호수 해안선의 길이를 판별

ISLANDS 테이블의 ID 및 이름 컬럼은 각 섬을 나타내며 섬 다각형 컬럼은 섬의 기하학을 저장합니다.

```
CREATE TABLE ISLANDS (id    integer,
                      name  varchar(32),
                      land  db2gse.ST_Polygon);
```

다음의 ODBC 프로그램에서는 ST_InteriorRingN 함수를 사용하여 각 섬 다각형에서 내부 링(호수)을 선스트링으로서 발췌합니다. 길이 함수에 의해 리턴되는 선스트링의 주계를 모두 합해 섬의 ID와 함께 표시합니다.

```
#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;

/* ODBC 환경 핸들 henv에 대한 메모리를 할당하고,
                  응용프로그램을 초기화합니다. */

  rc = SQLAllocEnv (&henv);
    if (rc != SQL_SUCCESS)
  {
    printf ("SQLAllocEnv failed with %d\n", rc);
    exit(0);
  }

/* henv 환경 내에 연결 핸들에 대한 메모리를 할당합니다. */

  rc = SQLAllocConnect (henv, &handle);
    if (rc != SQL_SUCCESS)
  {
    printf ("SQLAllocConnect failed with %d\n", rc);
    exit(0);
  }

/* ODBC 드라이버를 로드하고 데이터베이스, 사용자 및 암호에 의해
                식별되는 데이터 소스에 연결합니다. */

  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");

/* 메모리를 SQL문 핸들 island_cursor에 할당합니다. */

  rc = SQLAllocStmt (handle, &island_cursor);
  check_sql_err (handle, NULL, rc, "SQLAllocStmt");

/* island ID 및 lakes
   (내부 링)을 가져오기 위한 조회를 준비하고 실행합니다. */

  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");

/* island 테이블의 ID 컬럼을 변수 island_id에 바인드합니다.*/

  rc = SQLBindCol (island_cursor, 1, SQL_C_SLONG, &island_id, 0, &id_ind);
```

```
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

/* numinteriorrings(land)의 결과를 num_lakes 변수에 바인드합니다. */

  rc = SQLBindCol (island_cursor, 2, SQL_C_SLONG, &num_lakes, 0, &lake_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

/* 메모리를 SQL문 핸들 lake_cursor에 할당합니다. */

rc = SQLAllocStmt (handle, &lake_cursor);
  check_sql_err (handle, NULL, rc, "SQLAllocStmt");

/* 내부 링의 길이를 가져오기 위한 조회를 준비합니다. */

  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");

/* lake_number 변수를 첫번째 입력 매개변수로 바인드합니다. */

  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");

/* island_id를 두 번째 입력 매개변수로 바인드합니다. */

  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");

/* Length(db2gse.ST_InteriorRingN(land, cast (? as integer)))의 결과를
   변수 lake_perimeter에 바인드합니다. */

  rc = SQLBindCol (lake_cursor, 1, SQL_C_SLONG, &lake_perimeter, 0,
                   &length_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

  /* 외부 루프(Outer loop), island id 및 lakes (내부 링)의 수를 가져옵니다. */

  while (SQL_SUCCESS == rc)
  {
    /* island를 사전 추출합니다. */

    rc = SQLFetch (island_cursor);

    if (rc != SQL_NO_DATA)
    {
      check_sql_err (NULL, island_cursor, rc, "SQLFetch");

      /* 내부 루프(Inner loop), 이 island에 대해 모든
         lakes (내부 링)의 주계를 가져옵니다. */

      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);
      }
    }

/* Island id 및 이의 lakes에 대한 전체 주계를 표시합니다. */

    printf ("Island ID = %d,  Total lake perimeter = %d\n",
             island_id,total_perimeter);

  }

  SQLFreeStmt (lake_cursor, SQL_DROP);
  SQLFreeStmt (island_cursor, SQL_DROP);
        SQLDisconnect (handle);
        SQLFreeConnect (handle);
  SQLFreeEnv (henv);

  printf( "\nTest Complete ...\n" );

}

static void check_sql_err (SQLHDBC handle, SQLHSTMT  hstmt, LONG rc,
                           CHAR *str)
{

    SDWORD dbms_err = 0;
    SWORD  length;
    UCHAR   err_msg[SQL_MAX_MESSAGE_LENGTH], state[6];

    if (rc != SQL_SUCCESS)
    {
      SQLError (SQL_NULL_HENV, handle, hstmt, state, &dbms_err,
                err_msg, SQL_MAX_MESSAGE_LENGTH - 1, &length);
      printf ("%s ERROR (%d): DBMS code:%d, SQL state: %s, message: \n %s\n",
               str, rc, dbms_err, state, err_msg);

      if (handle)
      {
        SQLDisconnect (handle);
        SQLFreeConnect (handle);
      }
      exit(1);
    }
}
```

ST_Intersection

ST_Intersection은 두 개의 기하학 오브젝트를 취해 교차 세트를 기하학 오브젝트로서 리턴합니다.

구문

db2gse.ST_Intersection(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

db2gse.ST_Geometry

예

소방서장은 예상되는 위험 폐기물 오염원의 반경 이내에 교차되는 병원, 학교 및 사설 요양원 영역에 관련된 정보를 알고 있어야 합니다.

중요 영역은 다음의 CREATE TABLE문으로 작성된 테이블 SENSITIVE_AREAS에 저장됩니다. ZONE 컬럼은 각 중요 영역의 외형을 저장하는 다각형으로서 저장됩니다.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

위험 사이트는 다음의 CREATE TABLE문으로 작성되는 HAZARDOUS_SITES 테이블에 저장됩니다. 점으로 정의된 LOCATION 컬럼은 각 위험 사이트의 지리적 중심이 되는 위치를 저장합니다.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

버퍼 함수는 위험 폐기물 사이트 위치를 둘러싼 5마일 버퍼를 생성합니다. ST_Intersection 함수는 버퍼화된 위험 폐기물 사이트 다각형과 주요 영역의 교차점으로부터 다각형을 생성합니다. ST_Area 함수는 교차 다각형의 영역을 리턴하며 이는 SUM 함수에 의해 각 위험 사이트에 대해 요약됩니다. GROUP BY절은 위험 폐기물 site_ID별로 교차된 영역을 누적하도록 조회에 지시합니다.

```
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;
```

그림35에서 원은 위험 폐기물 사이트를 둘러싼 버퍼화된 다각형을 나타냅니다. 주요 영역 다각형이 있는 이 버퍼 다각형들이 교차되어 세 가지의 다른 다각형을 생성합니다. 왼쪽 상단 모서리에 있는 병원은 두 번 교차되는 반면에 오른쪽 하단 모서리에 있는 학교는 한 번만 교차됩니다.

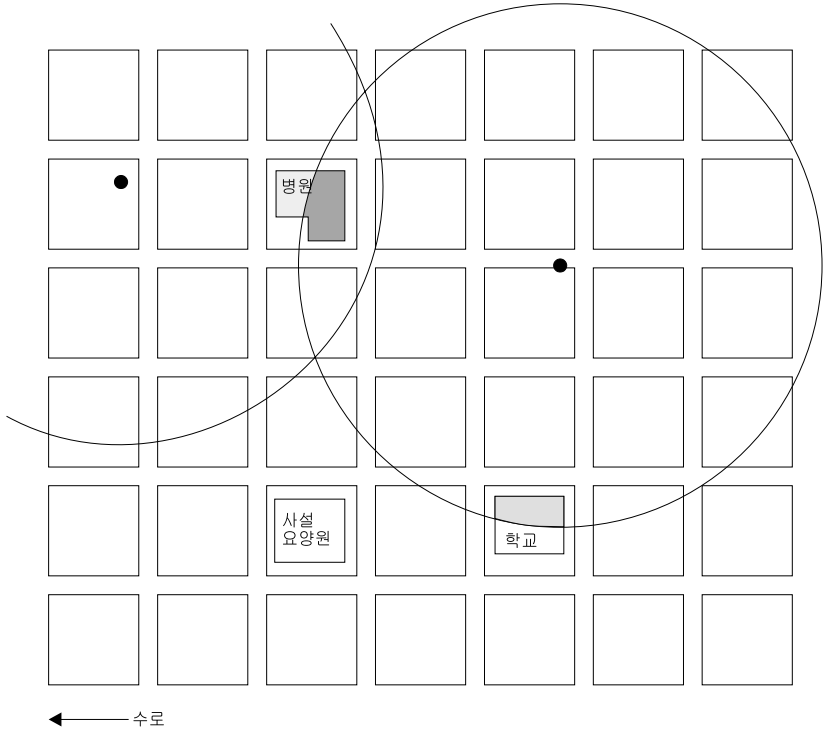


그림 35. ST_Intersection을 사용하여 각 빌딩에 있는 얼마나 넓은 영역이 위험 폐기물에 의해 영향을 받는지 판별

ST_Intersects

ST_Intersects는 두 개의 기하학을 취해, 두 기하학이 교차한 결과 세트가 비어 있지 않으면 1(TRUE)을 리턴합니다. 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_Intersects(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

소방서장은 위험 폐기물 사이트의 5마일 반지름 내에 있는 모든 주요 영역의 목록이 필요합니다.

중요 영역은 다음의 CREATE TABLE문으로 작성된 테이블 SENSITIVE_AREAS에 저장됩니다. ZONE 컬럼은 각 중요 영역의 외형을 저장하는 다각형으로서 저장됩니다.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

위험 사이트는 다음의 CREATE TABLE문으로 작성되는 HAZARDOUS_SITES 테이블에 저장됩니다. 점으로 정의된 LOCATION 컬럼은 각 위험 사이트의 지리적 중심이 되는 위치를 저장합니다.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);
```

조회에서는 주요 영역의 목록과 위험 사이트의 5마일 버퍼를 교차하는 주요 영역에 대한 위험 사이트 이름들을 나열합니다.

```
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_IsClosed는 선스트링 또는 선스트링을 취해 그것이 닫혀 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_IsClosed(c db2gse.ST_Curve)
db2gse.ST_IsClosed(mc db2gse.ST_MultiCurve)

리턴 유형

정수

예

다음의 CREATE TABLE 문은 하나의 선스트링 컬럼이 있는 CLOSED_LINESTRING 테이블을 작성합니다.

```
CREATE TABLE CLOSED_LINESTRING (ln1 db2gse.ST_LineString)
```

다음의 INSERT문은 CLOSED_LINESTRING 테이블에 두 개의 레코드를 삽입합니다. 첫번째 레코드는 닫힌 선스트링이 아니고 두 번째는 닫힌 선스트링입니다.

```
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)))
```

다음의 SELECT문 및 해당 결과 세트에서는 ST_IsClosed 함수의 결과를 보여 줍니다. 첫번째 행은 선스트링이 닫히지 않았기 때문에 0을 리턴하는 반면에 두 번째 행은 선스트링이 닫혀 있기 때문에 1을 리턴합니다.

```
SELECT db2gse.ST_IsClosed(ln1) "Is it closed" FROM CLOSED_LINESTRING

Is it closed
------------
0
1

  2 record(s) selected.
```

다음의 CREATE TABLE문은 하나의 다중 선스트링 컬럼이 있는 CLOSED_MULTILINESTRING 테이블을 작성합니다.

```
CREATE TABLE CLOSED_MULTILINESTRING (mln1 db2gse.ST_MultiLineString)
```

다음의 INSERT문은 CLOSED_MULTILINESTRING에 두 개의 레코드 즉, 닫히지 않은 다중 선스트링 레코드와 닫힌 또다른 다중 선스트링 레코드를 삽입합니다.

```
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)))
```

다음의 SELECT문 및 해당 결과 세트에서는 ST_IsClosed 함수의 결과를 보여 줍니다. 첫번째 행은 다중 선스트링이 닫혀 있지 않기 때문에 0을 리턴하지만 두 번째 행은 다중 선스트링이 닫힌 상태이기 때문에 1을 리턴합니다. 다중 선스트링의 선스트링 요소 모두가 닫힌 상태이면 다중 선스트링은 닫힙니다.

```
SELECT db2gse.ST_IsClosed(mln1) "Is it closed" FROM CLOSED_MULTILINESTRING

Is it closed
------------
0
1

  2 record(s) selected.
```

ST_IsEmpty

ST_IsEmpty는 기하학 오브젝트를 취해 오브젝트가 비어 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_IsEmpty(g db2gse.ST_Geometry)

리턴 유형

정수

예

다음의 CREATE TABLE문은 두 개의 컬럼이 있는 EMPTY_TEST 테이블을 작성합니다. GEOTYPE 컬럼은 G1 기하학 컬럼에 저장된 서브클래스의 데이터 유형을 저장합니다.

```
CREATE TABLE EMPTY_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

다음의 INSERT문은 기하학 서브클래스 점, 선스트링 및 다각형에 대해 두 레코드를 삽입합니다. 한 레코드는 비어 있고 한 레코드는 비어 있지 않습니다.

```
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)))
```

다음의 SELECT문과 해당 결과 세트에서는 GEOTYPE 컬럼으로부터의 기하학 유형과 ST_IsEmpty 함수의 결과를 보여 줍니다.

```
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_IsRing은 선스트링을 취해 그것이 링이면 즉, 선스트링이 닫힌 상태이고 단순하면 1(TRUE)을 리턴하고, 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_IsRing(c db2gse.ST_Curve)

리턴 유형

정수

예

다음의 CREATE TABLE 문은 LN1이라는 단일 선스트링 컬럼이 있는 RING_LINESTRING 테이블을 작성합니다.

```
CREATE TABLE RING_LINESTRING (ln1 db2gse.ST_LineString)
```

다음의 INSERT문은 LN1 컬럼에 세 개의 선스트링을 삽입합니다. 첫번째 행에는 닫혀 있지 않는 선스트링이 들어 있으므로 선스트링이 링이 아닙니다. 두 번째 행에는 닫힌 상태의 단순한 선스트링이 들어 있으므로 링입니다. 세 번째 행에는 닫힌 상태이지만 그 자신의 내부와 교차하기 때문에 단순하지 않은 선스트링이 들어 있으므로 링이 아닙니다.

```
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)))
```

다음의 SELECT문과 해당 결과 세트에서는 ST_IsRing 함수의 결과를 보여 줍니다. 첫번째 및 세 번째 행은 0을 리턴합니다. 이는 선스트링이 링이 아니기 때문이지만, 두 번째 행은 선스트링이 링이기 때문에 1을 리턴합니다.

```
SELECT db2gse.ST_IsRing(ln1) "Is it ring" FROM RING_LINESTRING

Is it ring
-----------
0
1
0

  3 record(s) selected.
```

ST_IsSimple

ST_IsSimple은 기하학 오브젝트를 취해 오브젝트가 단순하면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_IsSimple(g db2gse.ST_Geometry)

리턴 유형

정수

예

다음 CREATE TABLE문은 두 개의 컬럼이 있는 ISSIMPLE_TEST 테이블을 작성합니다. smallint인 PID 컬럼에는 각 행에 대한 고유 식별자가 들어 있습니다. G1 기하학 컬럼은 단순한 기하학 샘플과 단순하지 않은 기하학 컬럼을 저장합니다.

```
CREATE TABLE ISSIMPLE_TEST (pid smallint, g1 db2gse.ST_Geometry)
```

다음의 INSERT문은 ISSIMPLE_TEST 테이블에 두 개의 레코드를 삽입합니다. 첫번째는 이의 내부와 교차하지 않는 선스트링이기 때문에 단순합니다. 두 번째는 이의 내부와 교차하기 때문에 단순하지 않습니다.

```
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)))
```

다음의 SELECT문 및 해당 결과 세트에서는 ST_IsSimple 함수의 결과를 보여줍니다. 첫번째 레코드는 선스트링이 단순하기 때문에 1을 리턴하며, 두 번째 레코드는 선스트링이 단순하지 않기 때문에 0을 리턴합니다.

```
SELECT ST_IsSimple(g1)
FROM ISSIMPLE_TEST
```

```
g1
--------------
1
0
```

ST_IsValid

ST_IsValid는 ST_Geometry를 취해 그것이 유효하면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. DB2 Spatial Extender는 공간 데이터를 승인하기 전에 항상 유효성을 검사하기 때문에 DB2 데이터베이스에 삽입된 기하학은 항상 유효할 것입니다. 그러나, 다른 DBMS 벤더들은 입력의 유효성을 검사하지 않을 수도 있으므로 응용프로그램에서 대신 그 작업을 수행해야 합니다.

구문

db2gse.ST_IsValid(g db2gse.ST_Geometry)

리턴 유형

정수

예

valid_test 테이블은 컬럼 geotype 및 g1으로 작성됩니다. geotype 컬럼은 g1 기하학 컬럼에 저장된 기하학 서브클래스의 이름을 저장합니다.

```
CREATE TABLE valid_test (geotype varchar(20), g1 db2gse.ST_Geometry)
```

INSERT문은 샘플 서브클래스를 valid_test 테이블에 삽입합니다.

```
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))
)
```

다음의 SELECT문은 해당 geotype의 차원으로 geotype 컬럼에 저장된 서브클래스 이름을 나열합니다.

```
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_Length는 선스트링 또는 다중 선스트링을 취해 이의 길이를 리턴합니다.

구문

db2gse.ST_Length(c db2gse.ST_Curve)
db2gse.ST_Length(mc db2gse.ST_MultiCurve)

리턴 유형

Double

예

지역 생태학자는 도의 수로에서 연어 모집단의 이주 패턴을 연구 중입니다. 생태학자는 도를 관통하는 모든 하천 및 강 체계의 길이를 얻으려고 합니다.

다음의 CREATE TABLE문은 WATERWAYS 테이블을 작성합니다. ID 및 NAME 컬럼은 테이블에 저장된 각 하천 및 강 체계를 나타냅니다. 강 및 하천 체계는 자주 여러 선스트링의 총계가 되기 때문에 WATER 컬럼은 다중 선스트링입니다.

```
CREATE TABLE WATERWAYS (id integer, name varchar(128),
                                water      db2gse.ST_MultiLineString);
```

다음의 SELECT문은 ST_Length 함수를 사용하여 도에 있는 각 수로의 이름과 길이를 리턴합니다.

```
SELECT name, db2gse.ST_Length(water) "Length"
FROM WATERWAYS;
```

255 페이지의 그림36에서는 도 경계 내에 놓여 있는 강과 하천 체계를 표시합니다.

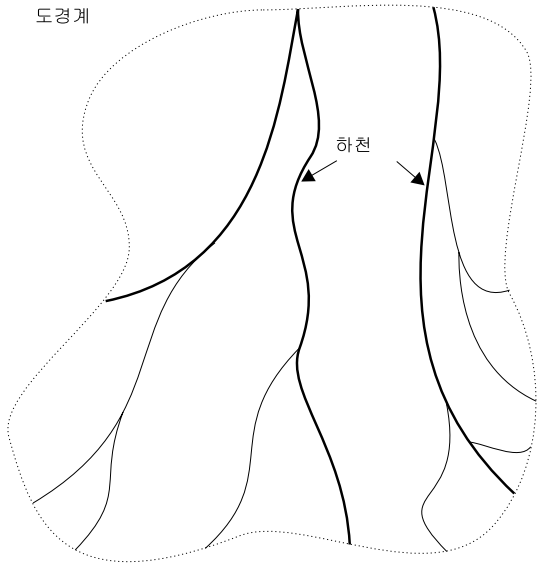


그림 36. ST_Length를 사용하여 도의 총 수로 길이를 판별

ST_LineFromText

ST_LineFromText는 선스트링 유형의 잘 알려진 텍스트 표현식 및 공간 참조 시스템 아이덴티티를 취해 선스트링을 리턴합니다.

구문

db2gse.ST_LineFromText(lineStringTaggedText Varchar(4000), cr db2gse.coordref)

리턴 유형

db2gse.ST_LineString

예

다음의 CREATE TABLE문은 하나의 LN1 선스트링 컬럼이 있는 LINESTRING_TEST 테이블을 작성합니다.

```
CREATE TABLE LINESTRING_TEST (ln1 db2gse.ST_LineString)
```

다음의 INSERT문은 ST_LineFromText 함수를 사용하여 LN1 컬럼에 선스트링을 삽입합니다.

```
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_LineFromWKB는 선스트링 유형의 잘 알려진 2진 표현식 및 공간 참조 시스템 아이덴티티를 취해 선스트링을 리턴합니다.

구문

db2gse.ST_LineFromWKB(WKBLineString Blob(1M), cr db2gse.coordref)

리턴 유형

db2gse.ST_LineString

예

다음의 코드 조각은 각 송수관선의 고유 ID, 크기 클래스 및 기하학을 가진 SEWERLINES 테이블을 상주시킵니다.

SEWERLINES 테이블은 세 개의 컬럼으로 작성됩니다. 첫번째 컬럼인 SEWER_ID는 각 송수관선을 고유하게 식별합니다. 정수 유형인 두 번째 컬럼 CLASS는 송수관선의 유형을 나타내며 이는 일반적으로 선의 용량과 연관됩니다. 선스트링 유형인 세 번째 컬럼 SEWER는 송수관선의 기하학을 저장합니다.

```
CREATE TABLE SEWERLINES (sewer_id   integer,
                         class  integer,
                         sewer  db2gse.ST_LineString);

/* sewer_id, 크기 class 및 sewer 선스트링을 상주시키기 위한 SQL문을
   작성합니다. 물음표는 매개변수 표시문자이며, 이는
   이는 런타임에 검색되는 sewer_id, class 및 sewer 기하학 값을
   지시합니다. */
strcpy (wkb_sql,"insert into sewerlines (sewer_id,class,sewer)
values (?,?, db2gse.ST_LineFromWKB (cast(? as blob(1m)), db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* 정수 sewer_id 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_id, 0, &pcbvalue1);

/* 정수 class 값을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = 0;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_class, 0, &pcbvalue2);

/* 형상을 세 번째 매개변수에 바인드합니다. */
pcbvalue3 = blob_len;
```

```
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, sewer_wkb, blob_len, &pcbvalue3);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

ST_MLineFromText

ST_MLineFromText는 다중 선스트링 유형의 잘 알려진 텍스트 표현식 및 공간 참조 시스템 아이덴티티를 취해 다중 선스트링을 리턴합니다.

구문

db2gse.ST_MLineFromText(multiLineStringTaggedText String, cr db2gse.coordref)

리턴 유형

db2gse.ST_MultiLineString

예

다음의 CREATE TABLE문은 MLINESTRING_TEST 테이블을 작성합니다. MLINESTRING_TEST에는 행을 고유하게 나타내는 GID smallint 컬럼과 ML1 다중 선스트링 컬럼이 있습니다.

```
CREATE TABLE ST_MLINESTRING_TEST (gid smallint, ml1 db2gse.ST_MultiLineString)
```

다음의 INSERT문은 ST_MLineFromText 함수로 다중 선스트링을 삽입합니다.

```
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_MLineFromWKB는 다중 선스트링 유형의 잘 알려진 2진 표현식 및 공간 참조 시스템 아이덴티티를 취해 다중 선스트링을 리턴합니다.

구문

db2gse.ST_MLineFromWKB(WKBMultiLineString Blob(1M), cr db2gse.coordref)

리턴 유형

db2gse.ST_MultiLineString

예

다음의 코드 조각은 고유 ID, 이름 및 수역 다중 선스트링을 가진 WATERWAYS 테이블을 데이터 상주시킵니다.

WATERWAYS 테이블은 테이블에 저장된 각 하천과 강 체계를 나타내는 ID 및 NAME 컬럼으로 작성됩니다. 강 및 하천 체계는 자주 여러 선스트링의 총계가 되기 때문에 WATER 컬럼은 다중 선스트링입니다.

```
CREATE TABLE WATERWAYS (id        integer,
                        name      varchar(128),
                        water     db2gse.ST_MultiLineString);

/* ID, name 및 다중 선스트링을 상주시키기 위한
   SQL문을 작성합니다. 물음표는 매개변수 표시문자이며, 이는
   런타임에 검색되는 ID, name 및 water 값을
   지시합니다. */
strcpy (shp_sql,"insert into WATERWAYS (id,name,water)
values (?,?, db2gse.ST_MLineFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* 정수 id 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
```

```
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);

/* varchar name 값을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, name_len, 0, &name, name_len, &pcbvalue2);

/* 형상을 세 번째 매개변수에 바인드합니다. */
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, water_shape, blob_len, &pcbvalue3);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

ST_MPointFromText

ST_MPointFromText는 다중점 유형의 잘 알려진 텍스트 표현식 및 공간 참조 시스템 아이덴티티를 취해 다중점을 리턴합니다.

구문

db2gse.ST_MPointFromText(multiPointTaggedText Varchar(4000), cr db2gse.coordref)

리턴 유형

db2gse.ST_MultiPoint

예

다음의 CREATE TABLE문은 하나의 MPT1 다중점 컬럼이 있는 MULTIPOINT_TEST 테이블을 작성합니다.

```
CREATE TABLE MULTIPOINT_TEST (mpt1 db2gse.ST_MultiPoint)
```

다음의 INSERT문은 ST_MPointFromText 컬럼을 사용하여 MPT1 컬럼에 다중점을 삽입합니다.

```
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_MPointFromWKB는 다중점 유형의 잘 알려진 2진 표현식 및 공간 참조 시스템 아이덴티티를 취해 다중점을 리턴합니다.

구문

db2gse.ST_MPointFromWKB(WKBMultiPoint Blob(1M), cr db2gse.coordref)

리턴 유형

db2gse.ST_MultiPoint

예

다음의 코드 조각은 SPECIES_SITINGS 테이블을 데이터 상주시킵니다.

SPECIES_SITINGS 테이블은 세 개의 컬럼으로 작성됩니다. SPECIES 및 GENUS 컬럼이 각 행을 고유하게 나타내고 SITINGS 다중점 컬럼은 종 구획(species sitings)의 위치를 저장합니다.

```
CREATE TABLE SPECIES_SITINGS (species  varchar(32),
                              genus varchar(32),
                              sitings  db2gse.ST_MultiPoint);

/* species, genus 및 sitings를 상주시키기 위한 SQL INSERT문을
   작성합니다. 물음표는 매개변수 표시문자이며, 이는
   런타임에 검색되는 species, genus 및 sitings 값을
   지시합니다. */
strcpy (wkb_sql,"insert into SPECIES_SITINGS (species,genus,sitings)
values (?,?, db2gse.ST_MPointFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* varchar species 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = species_len;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, species_len, 0, &species, species_len, &pcbvalue1);
/* varchar genus 값을 두 번째 매개변수에 바인드합니다. */
```

```
pcbvalue2 = genus_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, genus_len, 0, &name, genus_len, &pcbvalue2);

/* 형상을 세 번째 매개변수에 바인드합니다. */
pcbvalue3 = sitings_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, sitings_len, 0, sitings_wkb, sitings_len, &pcbvalue3);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

ST_MPolyFromText

ST_MPolyFromText는 다중 다각형 유형의 잘 알려진 텍스트 표현식 및 공간 참조 시스템 아이덴티티를 취해 다중 다각형을 리턴합니다.

구문

db2gse.ST_MPolyFromText(multiPolygonTaggedText Varchar(4000),cr db2gse.coordref)

리턴 유형

db2gse.ST_MultiPolygon

예

다음의 CREATE TABLE문은 하나의 MPL1 다중 다각형 컬럼이 있는 MULTIPOLYGON_TEST 테이블을 작성합니다.

```
CREATE TABLE MULTIPOLYGON_TEST (mpl1 db2gse.ST_MultiPolygon)
```

다음의 INSERT문은 ST_MPolyFromText 함수를 사용하여 MPL1 컬럼에 다중 다각형을 삽입합니다.

```
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_MPolyFromWKB는 다중 다각형 유형의 잘 알려진 2진 표현식 및 공간 참조 시스템 아이덴티티를 취해 다중 다각형을 리턴합니다.

구문

db2gse.ST_MPolyFromWKB(WKBMultiPolygon Blob(1M), cr db2gse.coordref)

리턴 유형

db2gse.ST_MultiPolygon

예

다음의 코드 조각은 LOTS 테이블을 데이터 상주시킵니다.

LOTS 테이블은 각 구획을 고유하게 나타내는 LOT_ID와 구획 선 기하학이 포함된 LOT 다중 다각형을 저장합니다.

```
CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );

/* lot_id 및 lot를 상주시키기 위한 SQL INSERT문을 작성합니다.
   물음표는 매개변수 표시문자이며, 이는 런타임에 검색되는 lot_id 및 lot
   값을 지시합니다. */
strcpy (wkb_sql,"insert into LOTS (lot_id,lot)
values (?, db2gse.ST_MPolyFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* lot_id 정수 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* lot 형상을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = lot_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, lot_len, 0, lot_wkb, lot_len, &pcbvalue2);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

ST_NumGeometries

ST_NumGeometries는 콜렉션을 취해 콜렉션 내의 기하학수를 리턴합니다.

구문

db2gse.ST_NumGeometries(g db2gse.ST_GeomCollection)

리턴 유형

정수

예

도시 공학자는 각 빌딩 위치와 연관된 별개 빌딩의 실제 수를 알고 있어야 합니다.

빌딩 위치는 다음의 CREATE TABLE문으로 작성되는 BUILDINGFOOTPRINTS 테이블에 저장됩니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                                 footprint   db2gse.ST_MultiPolygon);
```

다음의 SELECT문은 ST_NumGeometries 함수를 사용하여 각 빌딩을 고유하게 나타내는 BUILDING_ID와 각 위치에 있는 빌딩수를 나열합니다.

```
SELECT building_id, db2gse.ST_NumGeometries (footprint) "Number of buildings"
FROM BUILDINGFOOTPRINTS;
```

ST_NumInteriorRing

ST_NumInteriorRing은 다각형을 취해 이의 내부 링 수를 리턴합니다.

구문

db2gse.NumInteriorRing(p db2gse.ST_Polygon)

리턴 유형

정수

예

여러 남해 섬에 사는 조류 모집단을 연구하려는 조류학자는 특정 조류의 생태 구역이 청정 호수가 있는 섬에 제한됨을 알고 있습니다. 그러므로, 생태학자는 하나 이상의 호수가 있는 섬들을 알고자 할 것입니다.

다음의 CREATE TABLE문은 ISLANDS 테이블을 작성합니다. ISLANDS 테이블의 ID 및 NAME 컬럼은 각 섬을 나타내며 LAND 다각형 컬럼은 섬의 기하학을 저장합니다.

```
CREATE TABLE ISLANDS (id integer, name varchar(32), land db2gse.ST_Polygon);
```

내부 링이란 호수를 나타내기 때문에, ST_NumInteriorRing 함수는 적어도 하나의 내부 링이 있는 그러한 섬만을 나열하는 데 사용됩니다.

```
SELECT name FROM ISLANDS WHERE db2gse.ST_NumInteriorRing(land) > 0;
```

ST_NumPoints

ST_NumPoints는 선스트링을 취해 이의 점 수를 리턴합니다.

구문

db2gse.ST_NumPoints(l db2gse.ST_LineString)

리턴 유형

정수

예

다음의 CREATE TABLE문은 NUMPOINTS_TEST 테이블을 작성합니다. GEOTYPE 컬럼에는 G1 기하학 컬럼에 저장된 기하학 유형이 들어 있습니다.

```
CREATE TABLE NUMPOINTS_TEST (geotype varchar(12), g1 db2gse.ST_Geometry)
```

다음의 INSERT문은 선스트링을 삽입합니다.

```
INSERT INTO NUMPOINTS_TEST VALUES( linestring,
db2gse.ST_LineFromText('linestring (10.02 20.01, 23.73 21.92)',
db2gse.coordref()..srid(0)))
```

다음의 SELECT문 및 해당 결과 세트에서는 기하학 유형과 각각에 포함된 점 수를 나열합니다.

```
SELECT geotype, db2gse.ST_NumPoints(g1)
FROM NUMPOINTS_TEST

GEOTYPE      Number of points
------------ ----------------
ST_linestring          2
  1 record(s) selected.
```

ST_OrderingEquals

ST_OrderingEquals는 두 개의 기하학을 비교하여 기하학이 같으면서 좌표가 동일한 순서도 되어 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_OrderingEquals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

다음의 CREATE TABLE문은 두 개의 선스트링 컬럼 L1 및 L2가 있는 LINESTRING_TEST 테이블을 작성합니다.

```
CREATE TABLE LINESTRING_TEST (lid integer, l1 db2gse.ST_LineString,
                    l2 db2gse.ST_LineString);
```

다음의 INSERT문은 내용도 같고 좌표 순서도 동일한 L1 및 L2로 두 선스트링을 삽입합니다.

```
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)));
```

다음의 INSERT문은 내용은 같지만 좌표 순서가 동일하지 않은 L1 및 L2로 두 선스트링을 삽입합니다.

```
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)));
```

다음의 SELECT문과 해당 결과 세트에서는 좌표 순서에 상관없이 ST_Equals 함수가 어떻게 1(TRUE)을 리턴하는지 보여줍니다. 두 기하학이 같지 않으며 동일한 좌표 순서를 가지는 경우, ST_OrderingEquals 함수는 0 (FALSE)를 리턴합니다.

```
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_Overlaps는 두 개의 기하학 오브젝트를 취해 오브젝트의 교차 결과가 동일한 차원의 기하학 오브젝트이지만 서로의 소스 오브젝트에 일치하지 않으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_Overlaps(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

도 감독자에게는 반경 5마일 내에 주요 지역이 있는 위험 폐기물 사이트 목록이 필요합니다.

다음의 CREATE TABLE문은 SENSITIVE_AREAS 테이블을 작성합니다. SENSITIVE_AREAS 테이블에는 공공빌딩의 다각형 기하학을 저장하는 존 컬럼 외에도 위협적인 공공빌딩을 서술하는 여러 컬럼들이 들어 있습니다.

```
CREATE TABLE SENSITIVE_AREAS (id       integer,
                              name     varchar(128),
                              size     float,
                              type     varchar(10),
                              zone     db2gse.ST_Polygon);
```

HAZARDOUS_SITES 테이블은 SITE_ID 및 NAME 컬럼에 사이트의 아이덴티티를 저장하는 반면에, 각 사이트의 실제 지리적 위치는 LOCATION 점 컬럼에 저장됩니다.

```
CREATE TABLE HAZARDOUS_SITES (site_id  integer,
                              name     varchar(128),
                              location db2gse.ST_Point);
```

다음의 SELECT에서 SENSITIVE_AREAS 및 HAZARDOUS_SITES 테이블이 ST_Overlaps 함수로 결합됩니다. 이는 존 다각형이 버퍼화된 반경 5마일의 HAZARDOUS_SITES 위치 점과 겹치는 모든 SENSITIVE_AREAS 테이블의 모든 행에 대해 1(TRUE)을 리턴합니다.

```
SELECT hs.name
FROM HAZARDOUS_SITES hs, SENSITIVE_AREAS sa
WHERE db2gse.ST_Overlaps (buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

그림37에서 병원과 학교가 도의 두 위험 폐기물 사이트의 5마일 반경 이내에 겹치는 반면에 사설 요양소는 겹치지 않습니다.

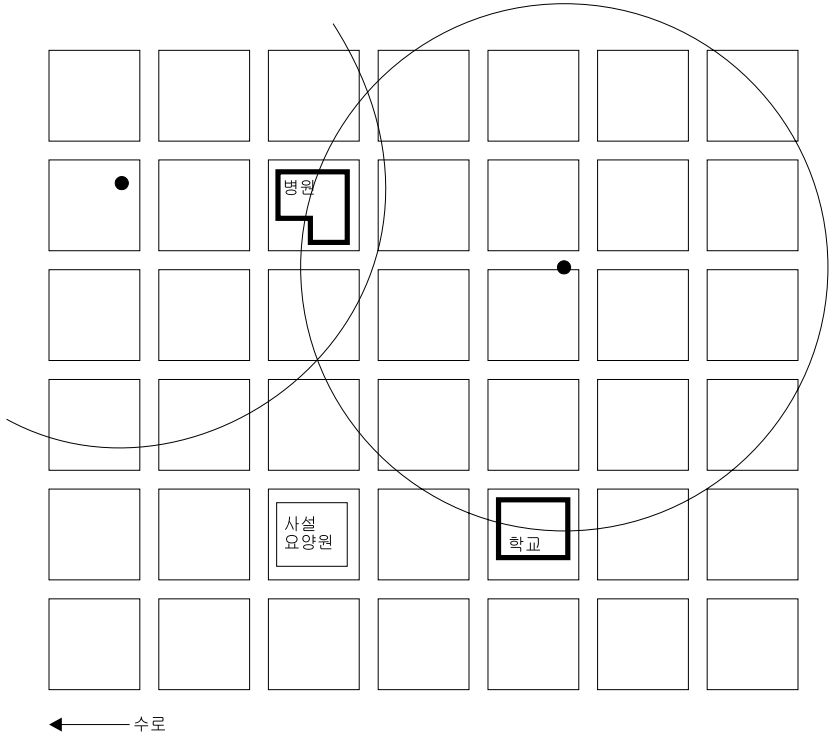


그림 37. ST_Overlaps를 사용하여 적어도 부분적으로 위험 폐기물 영역 내에 있는 빌딩을 판별

ST_Perimeter

ST_Perimeter는 ST_Surface의 주계를 리턴합니다.

구문

db2gse.ST_Perimeter(s db2gse.ST_Surface)
db2gse.ST_Perimeter(ms db2gse.ST_MultiSurface)

리턴 유형

Double

예

해안선 조류를 연구하는 생태학자는 특정 영역 내의 호수에 대한 해안선을 판별해야 합니다. 호수는 다음의 CREATE TABLE문으로 작성된 WATERBODIES 테이블에 다중 다각형으로서 저장됩니다.

```
CREATE TABLE WATERBODIES (wbid  integer,
                          waterbody  db2gse.ST_MultiPolygon);
```

다음의 SELECT문에서 ST_Perimeter 함수는 각 수역 본체를 둘러싼 주계를 리턴하는 반면에, SUM 함수는 주계를 누적하여 전체 합계를 리턴합니다.

```
SELECT SUM(db2gse.ST_Perimeter(waterbody))
FROM waterbodies;
```

ST_PointFromText

ST_PointFromText는 점 유형의 잘 알려진 텍스트 표현식 및 공간 참조 시스템 아이덴티티를 취해 점을 리턴합니다.

구문

db2gse.ST_PointFromText(pointTaggedText Varchar(4000), cr db2gse.coordref)

리턴 유형

db2gse.ST_Point

예

다음의 CREATE TABLE문은 하나의 PT1 점 컬럼이 있는 POINT_TEST 테이블을 작성합니다.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

INSERT문이 PT1 컬럼에 점을 삽입하기 전에 ST_PointFromText 함수는 점 텍스트 좌표를 점 형식으로 변환합니다.

```
INSERT INTO POINT_TEST VALUES (
 db2gse.ST_PointFromText ('point(10.01 20.03)', db2gse.coordref()..srid(0)))
```

ST_PointFromWKB

ST_PointFromWKB는 점 유형의 잘 알려진 2진 표현식 및 공간 참조 시스템 아이덴티티를 취해 점을 리턴합니다.

구문

db2gse.ST_PointFromWKB(WKBPoint Blob(1M), srs SRID)

리턴 유형

db2gse.ST_Point

예

다음의 코드 조각은 HAZARDOUS_SITES 테이블을 데이터 상주시킵니다.

위험 사이트는 다음의 CREATE TABLE문으로 작성되는 HAZARDOUS_SITES 테이블에 저장됩니다. 점으로 정의된 LOCATION 컬럼은 각 위험 사이트의 지리적 중심이 되는 위치를 저장합니다.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  db2gse.ST_Point);

/* site_id, name 및 location을 상주시키기 위한 SQL INSERT문을
   작성합니다. 물음표는 매개변수 표시문자이며, 이는
   런타임에 검색되는 site_id, name 및 location 값을 지시합니다. */
strcpy (wkb_sql,"insert into HAZARDOUS_SITES (site_id, name, location)
values (?,?, db2gse.ST_PointFromWKB(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* site_id 정수 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);

/* name varchar 값을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* location 형상을 세 번째 매개변수에 바인드합니다. */
```

```
pcbvalue3 = location_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, location_len, 0, location_wkb, location_len, &pcbvalue3);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

ST_Point

ST_Point는 X 좌표, Y 좌표 및 공간 참조로 주어진 ST_Point를 리턴합니다.

구문

db2gse.ST_Point(X Double, Y Double, srs SRID)

리턴 유형

db2gse.ST_Point

예

다음의 CREATE TABLE문은 하나의 PT1 점 컬럼이 있는 POINT_TEST 테이블을 작성합니다.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

ST_Point 함수는 INSERT문이 PT1 컬럼에 이를 삽입하기 전에 점 좌표를 점 기하학으로 변환합니다.

```
INSERT INTO point_test VALUES(
    db2gse.ST_Point(10.01,20.03, db2gse.coordref()..srid(0))
)
```

ST_PointN

ST_PointN은 선스트링과 정수 색인을 취해 선스트링의 경로에 있는 N 번째 정점인 점을 리턴합니다.

구문

db2gse.ST_PointN(1 db2gse.ST_Curve, n Integer)

리턴 유형

db2gse.ST_Point

예

다음의 CREATE TABLE문은 각 행을 고유하게 나타내는 GID 컬럼과 LN1 선스트링 컬럼 두 개가 있는 POINTN_TEST 테이블을 작성합니다.

```
CREATE TABLE POINTN_TEST (gid integer, ln1 db2gse.ST_LineString)
```

다음의 INSERT문은 두 선스트링값을 삽입합니다. 첫번째 선스트링에는 Z 좌표나 치수가 들어 있지 않으며 두 번째 선스트링에는 둘 모두 들어 있습니다.

```
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)))
```

다음의 SELECT문 및 해당 결과 세트에서는 GID 컬럼과 각 선스트링의 두 번째 정점을 나열합니다. 첫번째 행의 결과는 Z 좌표 또는 치수가 없는 점이고 두 번째 행 결과는 Z 좌표 또는 치수가 있는 점입니다. ST_PointN 함수는 Z 좌표 또는 특정 단위가 소스 선스트링에 존재하는 경우에는 Z 좌표 또는 치수와 함께 점을 리턴합니다.

```
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_PointOnSurface는 다각형 또는 다중 다각형 둘 모두를 취해 ST_Point를 리턴합니다.

구문

db2gse.ST_PointOnSurface(s db2gse.ST_Surface)
db2gse.ST_PointOnSurface(ms db2gse.ST_MultiSurface)

리턴 유형

db2gse.ST_Point

예

도시 공학자는 각 빌딩 위치에 대한 레이블 점을 작성해야 합니다.

빌딩 위치는 다음의 CREATE TABLE문으로 작성되는 BUILDINGFOOTPRINTS 테이블에 저장됩니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                                 footprint   db2gse.ST_MultiPolygon);
```

ST_PointOnSurface 함수는 빌딩 위치의 표면에 있게될 점을 생성합니다. ST_PointOnSurface 함수는 AsBinaryShape 함수가 응용프로그램이 사용할 1 MB 문자열에 배정된 형상으로 변환하는 점을 리턴합니다.

```
SELECT CAST(db2gse.AsBinaryShape(db2gse.ST_PointOnSurface(footprint)) as blob(1m))
FROM BUILDINGFOOTPRINTS;
```

ST_PolyFromText

ST_PolyFromText는 다각형 유형의 잘 알려진 텍스트 표현식 및 공간 참조 시스템 아이덴티티를 취해 다각형을 리턴합니다.

구문

db2gse.ST_PolyFromText(polygonTaggedText Varchar(4000), cr db2gse.coordref)

리턴 유형

db2gse.ST_Polygon

예

다음의 CREATE TABLE문은 하나의 다각형 컬럼이 있는 POLYGON_TEST 테이블을 작성합니다.

```
CREATE TABLE POLYGON_TEST (pl1 db2gse.ST_Polygon)
```

다음의 INSERT문은 ST_PolyFromText 함수를 사용하여 다각형 컬럼에 다각형을 삽입합니다.

```
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_PolyFromWKB는 다각형 유형의 잘 알려진 2진 표현식 및 공간 참조 시스템 아이덴티티를 취해 다각형을 리턴합니다.

구문

db2gse.ST_PolyFromWKB(WKBPolygon Blob(1M), SRID Integer)

리턴 유형

db2gse.ST_Polygon

예

다음의 코드 조각은 SENSITIVE_AREAS 테이블을 데이터 상주시킵니다.

SENSITIVE_AREAS 테이블에는 공공빌딩의 다각형 기하학을 저장하는 존 컬럼 외에도 위협적인 공공빌딩을 서술하는 여러 컬럼들이 들어 있습니다.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);


/* name, size, type 및 zone을 상주시키기 위한 SQL INSERT문을
   작성합니다. 물음표는 매개변수 표시문자이며, 이는 런타임에 검색되는
   id, name, size, type 및 zone 값을 지시합니다. */
strcpy (shp_wkb,"insert into SENSITIVE_AREAS (id, name, size, type, zone)
values (?,?,?,?, db2gse.ST_PolyFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */
rc = SQLAllocStmt (handle, &hstmt);

/* 실행을 위한 SQL문을 준비합니다. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* id 정수 값을 첫번째 매개변수에 바인드합니다. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);

/* name varchar 값을 두 번째 매개변수에 바인드합니다. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);
```

```
/* size float를 세 번째 매개변수에 바인드합니다. */
pcbvalue3 = 0;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_FLOAT,
     SQL_REAL, 0, 0, &size, 0, &pcbvalue3);

/* type varchar을 네 번째 매개변수에 바인드합니다. */
pcbvalue4 = type_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_VARCHAR, type_len, 0, type, type_len, &pcbvalue4);

/* zone 다각형을 다섯 번째 매개변수에 바인드합니다. */
pcbvalue5 = zone_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, zone_len, 0, zone_wkb, zone_len, &pcbvalue5);

/* INSERT문을 실행합니다. */
rc = SQLExecute (hstmt);
```

ST_Polygon

ST_Polygon은 ST_LineString 및 공간 참조 시스템 식별자로부터 ST_Polygon을 생성합니다.

구문

db2gse.ST_Polygon(l db2gse.ST_LineString, cr db2gse.coordref)

리턴 유형

db2gse.ST_Polygon

예

다음의 CREATE TABLE문은 하나의 P1 컬럼이 있는 POLYGON_TEST 테이블을 작성합니다.

```
CREATE TABLE POLYGON_TEST (p1 db2gse.ST_polygon)
```

다음의 INSERT문은 링(닫혀 있는 선스트링 및 단순한 선스트링 모두)을 다각형으로 변환하고, ST_Polygon 함수 내의 ST_LineFromText 함수를 사용하여 P1 컬럼으로 이를 삽입합니다.

```
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_Relate는 두 기하학을 비교하여 기하학들이 DE-9IM 패턴 행렬 문자열에 의해 지정된 조건을 충족하면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)이 리턴됩니다. 149 페이지의 『술어 함수』에서 DE-9IM 패턴 행렬에 대한 자세한 내용을 참조하십시오.

구문

db2gse.ST_Relate(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry, patternMatrix String)

리턴 유형

정수

예

DE-9IM 패턴 행렬은 기하학을 비교하기 위한 장치입니다. 여러 가지 유형의 그와 같은 행렬들이 있습니다. 예를 들어, *equals* 패턴 행렬은 두 기하학이 동일한지를 알려 줍니다.

이 예에서 표56에 표시된 equals 패턴 행렬은 왼쪽에서 오른쪽으로 위에서 아래쪽으로 문자열(『T*F**FFF*』)에 읽혀집니다.

표 56. Equals 패턴 행렬

		b		
		내부	**경계**	**외부**
a	**내부**	T	*	F
	경계	*	*	F
	외부	F	F	*

테이블 RELATE_TEST는 다음의 CREATE TABLE문으로 작성됩니다.

```
CREATE TABLE RELATE_TEST (rid integer, g1 db2gse.ST_Geometry,
g2 db2gse.ST_Geometry, g3 db2gse.ST_Geometry);
```

다음의 INSERT문은 RELATE_TEST 테이블에 샘플 서브클래스를 삽입합니다.

```
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)
)
```

다음의 SELECT문과 해당 결과 세트는 그 geotype의 차원으로 GEOTYPE 컬럼에 저장된 서브클래스 이름을 나열합니다.

```
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_SRID는 기하학 오브젝트를 취해 이의 공간 참조 시스템 아이덴티티를 리턴합니다.

구문

db2gse.ST_SRID(g1 db2gse.ST_Geometry)

리턴 유형

정수

예

DB2 Spatial Extender를 설치할 때 SPATIAL_REFERENCES 테이블이 작성됩니다. 기하학이 작성될 때 해당 기하학의 SRID가 SPATIAL_REFERENCES 테이블에 입력됩니다. ST_SRID 함수는 그 항목의 값을 리턴합니다.

예를 들어, 기하학 유형은 CREATE TABLE문에 사용됩니다.

```
CREATE TABLE SRID_TEST(g1 db2gse.ST_Geometry)
```

다음 INSERT문에서는 좌표 10.01,50.76에 놓인 점 기하학이 기하학 컬럼 G1에 삽입됩니다. ST_PointFromText 함수가 점 기하학을 작성할 때, srid 값 1을 지정합니다.

```
INSERT INTO SRID_TEST
VALUES (db2gse.ST_PointFromText('point(10.01 50.76)', db2gse.coordref()..srid(0)))
```

ST_SRID 함수는 다음의 SELECT문과 해당 결과 세트에서 설명한대로 방금 입력된 기하학의 공간 참조 시스템 아이덴티티를 리턴합니다.

```
SELECT db2gse.ST_SRID(g1) FROM SRID_TEST

g1
--------------
1
```

ST_StartPoint

ST_StartPoint는 선스트링을 취해 선스트링의 첫번째 점인 점 하나를 리턴합니다.

구문

db2gse.ST_StartPoint(c db2gse.ST_Curve)

리턴 유형

db2gse.ST_Point

예

다음의 CREATE TABLE문은 STARTPOINT_TEST 테이블을 작성합니다. STARTPOINT_TEST에는 테이블 행을 고유하게 나타내는 GID 정수 컬럼과 LN1 선스트링 컬럼 두 개가 있습니다.

```
CREATE TABLE STARTPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

다음의 INSERT문은 LN1 컬럼으로 선스트링을 삽입합니다. 첫번째 선스트링에는 Z 좌표나 치수가 들어 있지 않으며 두 번째 선스트링에는 둘 모두 들어 있습니다.

```
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)))
```

다음의 SELECT문 및 해당 결과 세트에서는 ST_StartPoint 함수가 각 선스트링의 첫번째 점을 발췌하는 방법을 보여 줍니다. ST_AsText 함수는 점을 이의 텍스트 형식으로 변환합니다. 목록의 첫번째 점에는 Z 좌표 또는 치수가 없는 반면에, 두 번째 점에는 소스 선스트링에 Z 좌표 또는 치수가 있기 때문에 둘 모두 있습니다.

```
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_SymmetricDiff는 두 개의 기하학 오브젝트를 취해 소스 오브젝트와 대칭적인 차이가 있는 기하학 오브젝트를 리턴합니다.

구문

db2gse.ST_SymmetricDiff(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

db2gse.ST_Geometry

예

도 감독자는 주요 영역과 5마일 위험 사이트 반경이 교차되지 않는지 판별해야 합니다.

다음의 CREATE TABLE문은 위협적인 공공 빌딩을 서술하는 여러 컬럼이 들어 있는 SENSITIVE_AREAS 테이블을 작성합니다. SENSITIVE_AREAS 테이블에는 공공빌딩의 다각형 기하학을 저장하는 ZONE 컬럼도 들어 있습니다.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

다음의 CREATE TABLE문은 HAZARDOUS_SITES 테이블을 작성하는데 이는 SITE_ID 및 NAME 컬럼에 사이트의 아이덴티티를 저장하는 반면에 각 사이트의 실제 지리적 위치는 LOCATION 점 컬럼에 저장됩니다.

```
CREATE TABLE HAZARDOUS_SITES (site_id   integer,
                              name      varchar(128),
                              location  point);
```

ST_Buffer 함수는 위험 폐기물 사이트 위치를 둘러싼 5마일 버퍼를 생성합니다. ST_SymmetricDiff 함수는 버퍼화된 위험 폐기물 사이트 다각형 및 주요 영역의 교차점으로부터 다각형을 생성합니다. ST_Area 함수는 각 위험 사이트에 대한 교차 다각형의 영역을 리턴합니다.

```
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
```

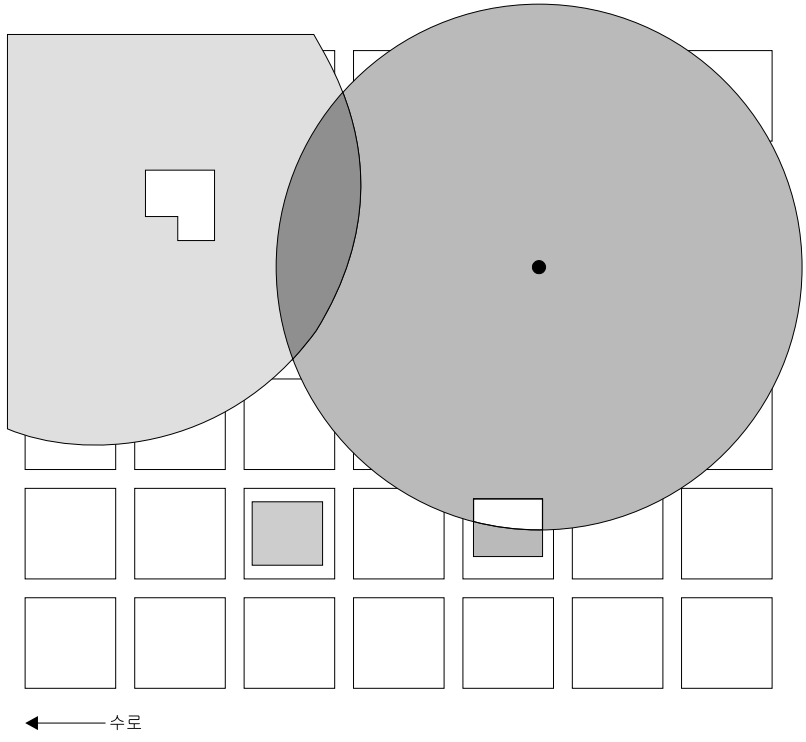


그림 38. ST_SymmetricDiff를 사용하여 주요 영역(사람이 거주하는 빌딩)이 포함되지 않은 위험 폐기물 영역을 판별

그림38에서 위험 폐기물 사이트와 주요 영역의 대칭적인 차이는 교차된 영역을 빼기한 결과입니다.

ST_Touches

ST_Touches는 두 기하학에 공통적인 점 중 어느 것도 두 기하학의 내부와 교차하지 않으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다. 적어도 하나의 기하학이 선스트링, 다각형, 다중 선스트링 또는 다중 다각형이어야 합니다.

구문

db2gse.ST_Touches(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

GIS 전문가는 송수관의 끝점이 다른 송수관과 교차하는 모든 송수관선의 목록을 제공해야 합니다.

다음 CREATE TABLE문은 세 개의 컬럼이 있는 SEWERLINES 테이블을 작성합니다. 첫번째 컬럼인 SEWER_ID는 각 송수관선을 고유하게 식별합니다. 정수 유형인 두 번째 컬럼 CLASS는 송수관선의 유형을 나타내며 이는 일반적으로 선의 용량과 연관됩니다. 선스트링 유형인 세 번째 컬럼 SEWER는 송수관선의 기하학을 저장합니다.

```
CREATE TABLE SEWERLINES (sewer_id integer, class integer, sewer
       db2gse.ST_LineString);
```

다음의 SELECT문은 서로 만나는 SEWER_IDS의 순서화된 목록을 리턴합니다.

```
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_Transform은 현재 기하학이 지정된 공간 참조 시스템이 아닌 다른 공간 참조 시스템에 기하학을 지정합니다.

구문

db2gse.ST_Transform(g db2gse.ST_Geometry, cr db2gse.coordref)

리턴 유형

db2gse.ST_Geometry

예

다음의 CREATE TABLE문은 두 개의 선스트링 컬럼 L1 및 L2가 있는 TRANSFORM_TEST 테이블을 작성합니다.

```
CREATE TABLE TRANSFORM_TEST (tid integer, l1 db2gse.ST_LineString, l2
       db2gse.ST_LineString)
```

다음의 INSERT문은 선스트링을 SRID 102인 l1에 삽입합니다.

```
INSERT INTO TRANSFORM_TEST VALUES (1, db2gse.ST_LineFromText
            ('linestring(10.01 40.43,92.32 29.89)',
            db2gse.coordref()..srid(102)),NULL)
```

ST_Transform 함수는 SRID 102에 지정된 좌표 참조로부터 SRID 105에 지정된 좌표 참조로 L1 선스트링을 변환합니다. 다음의 UPDATE문은 변환된 선스트링을 컬럼 12에 저장합니다.

```
UPDATE TRANSFORM_TEST SET l2 = db2gse.ST_Transform(l1,
       db2gse.coordref()..srid(105))
```

ST_Union

ST_Union은 두 개의 기하학 오브젝트를 취해 소스 오브젝트를 통합한 기하학 오브젝트를 리턴합니다.

구문

db2gse.ST_Union(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

db2gse.ST_Geometry

예

다음의 CREATE TABLE문은 위협적인 공공 빌딩을 서술하는 여러 컬럼이 들어 있는 SENSITIVE_AREAS 테이블을 작성합니다. SENSITIVE_AREAS 테이블에는 공공빌딩의 다각형 기하학을 저장하는 ZONE 컬럼도 들어 있습니다.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                              size      float,
                              type      varchar(10),
                              zone      db2gse.ST_Polygon);
```

다음의 CREATE TABLE문은 HAZARDOUS_SITES 테이블을 작성하는데 이는 SITE_ID 및 NAME 컬럼에 사이트의 아이덴티티를 저장합니다. 각 사이트의 실제 지리적 위치는 LOCATION 점 컬럼에 저장됩니다.

```
CREATE TABLE HAZARDOUS_SITES (site_id integer, name varchar(128),
                              location  db2gse.ST_Point);
```

다음의 SELECT문에서는 ST_Buffer 함수를 사용하여 위험 폐기물 사이트 위치를 둘러싼 5마일 버퍼를 생성합니다. ST_Union 함수는 버퍼화된 위험 폐기물 사이트 다각형과 주요 영역을 통합하여 다각형을 생성합니다. ST_Area 함수는 통합된 다각형 영역을 리턴합니다.

```
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_Within은 두 개의 기하학 오브젝트를 취해 첫번째 오브젝트가 완벽하게 두 번째 오브젝트 내에 있으면 1(TRUE)을 리턴하고 그렇지 않으면 0(FALSE)을 리턴합니다.

구문

db2gse.ST_Within(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

리턴 유형

정수

예

아래의 예에서는 두 개의 테이블이 작성됩니다. 첫번째 테이블인 BUILDINGFOOTPRINTS에는 도시의 빌딩 위치가 포함됩니다. 두 번째 테이블인 LOTS에는 도시 구획이 포함됩니다. 도시 공학자는 모든 빌딩 위치가 이의 구획 안에 완벽하게 있는지 확인하려 합니다.

양 테이블에서 다중 다각형 데이터 유형은 빌딩 위치와 구획의 기하학을 저장합니다. 데이터베이스 설계자는 강과 같은 자연적인 기능에 의해 구획이 결합 해제될 수 있고 빌딩 위치가 자주 여러 빌딩들로 구성될 수 있기 때문에 두 기능에 대해 다중 다각형을 선택했습니다.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                                 footprint   db2gse.ST_MultiPolygon);


CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

다음의 SELECT문을 사용하여 도시 공학자는 먼저 한 구획 내에 완전히 포함되지 않은 빌딩들을 선택합니다.

```
SELECT building_id
  FROM BUILDINGFOOTPRINTS, LOTS
 WHERE db2gse.ST_Within(footprint,lot) = 0;
```

첫번째 조회에서 구획 다각형의 외부에 위치한 모든 BUILDING_ID의 목록을 제공할지라도, 이는 나머지가 이들에 지정된 올바른 lot_id를 가지고 있는지 판별하지 않습니다. 이 두 번째 SELECT문에서는 BUILDINGFOOTPRINTS 테이블의 LOT_ID 컬럼에 대해 데이터 무결성 점검을 수행합니다.

```
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_WKBToSQL은 잘 알려진 2진 표현식으로 주어진 ST_Geometry 값을 구축합니다. SRID값 0은 자동으로 사용됩니다.

구문

db2gse.ST_WKBToSQL(WKBGeometry Blob(1M))

리턴 유형

db2gse.ST_Geometry

예

다음의 CREATE TABLE문은 각 구획을 고유하게 식별하는 LOT_ID 컬럼 및 각 구획의 기하학이 들어 있는 LOT 다각형 컬럼을 포함하는 LOTS 테이블을 작성합니다.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

다음의 C 코드 조각에는 LOTS 테이블에 데이터를 삽입하는 DB2 Spatial Extender SQL 함수와 함께 삽입된 ODBC 함수가 들어 있습니다.

ST_WKBToSQL 함수는 WKB 표현식을 DB2 Spatial Extender 기하학으로 변환합니다. 전체 INSERT문이 wkb_sql char 문자열로 복사됩니다. INSERT문에는 LOT_ID 데이터와 LOT 데이터를 동적으로 허용하는 매개변수 표시문자가 들어 있습니다.

```
/* lot ID 및 lot 다중 다각형을 상주시키기 위한
   SQL INSERT문을 작성합니다. 물음표는 매개변수 표시문자이며,
   이는 런타임에서 검색되는 lot_id 및 lot 값을
   지시합니다. */

strcpy (wkb_sql,"insert into lots (lot_id, lot)
  values(?, db2gse.ST_WKBToSQL(cast(? as blob(1m))))");

/* SQL문 핸들에 대한 메모리를 할당하고,
   명령문 핸들을 연결 핸들과 연결합니다. */

rc = SQLAllocStmt (handle, &hstmt);
```

```
/* 실행을 위한 SQL문을 준비합니다. */

rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* 정수 키 값을 첫번째 매개변수에 바인드합니다. */

pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* 형상을 두 번째 매개변수에 바인드합니다. */

pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* INSERT문을 실행합니다. */

rc = SQLExecute (hstmt);
```

ST_WKTToSQL

ST_WKTToSQL은 잘 알려진 텍스트 표현식으로 주어진 ST_Geometry 값을 구축합니다. SRID값 0은 자동으로 사용됩니다.

구문

db2gse.ST_WKTToSQL(geometryTaggedText Varchar(4000))

리턴 유형

db2gse.ST_Geometry

예

다음의 CREATE TABLE문은 각 행을 고유하게 나타내는 정수 유형의 GID 컬럼과 기하학을 저장하는 G1 컬럼이 있는 GEOMETRY_TEST 테이블을 작성합니다.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

다음의 INSERT문은 GEOMETRY_TEST 테이블의 GID 및 G1 컬럼에 데이터를 삽입합니다. ST_WKTToSQL 함수는 각 기하학의 텍스트 표현식을 이의 해당 DB2 Spatial Extender 인스턴스화 가능 서브클래스로 변환합니다.

```
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_X는 점을 취해 이의 X 좌표를 리턴합니다.

구문

ST_X(p ST_Point)

리턴 유형

Double

예

다음의 CREATE TABLE문은 행을 고유하게 나타내는 GID 컬럼과 PT1 점 컬럼이 있는 X_TEST 테이블을 작성합니다.

```
CREATE TABLE X_TEST (gid integer, pt1 db2gse.ST_Point)
```

다음의 INSERT문은 두 행을 삽입합니다. 하나는 Z 좌표나 치수가 없는 점이고, 다른 컬럼에는 Z 좌표와 치수 둘 모두 있습니다.

```
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)))
```

다음의 SELECT문 및 해당 결과 세트에서는 GID 컬럼 및 점의 배정밀도 X 좌표를 나열합니다.

```
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_Y는 점을 취해 이의 Y 좌표를 리턴합니다.

구문

db2gse.ST_Y(p db2gse.ST_Point)

리턴 유형

Double

예

다음의 CREATE TABLE문은 행을 고유하게 나타내는 GID 컬럼과 PT1 점 컬럼이 있는 Y_TEST 테이블을 작성합니다.

```
CREATE TABLE Y_TEST (gid integer, pt1 db2gse.ST_Point)
```

INSERT문은 두 행을 삽입합니다. 하나는 Z 좌표나 치수가 없는 점이고, 다른 컬럼에는 Z 좌표와 치수 둘 모두 있습니다.

```
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)))
```

다음의 SELECT문 및 해당 결과 세트에서는 GID 컬럼 및 점의 배정밀도 Y 좌표를 나열합니다.

```
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

Z는 점을 취해 이의 Z 좌표를 리턴합니다.

구문

Z(p db2gse.ST_Point)

리턴 유형

Double

예

다음의 CREATE TABLE문은 행을 고유하게 나타내는 GID 컬럼과 PT1 점 컬럼이 있는 Z_TEST 테이블을 작성합니다.

```
CREATE TABLE Z_TEST (gid integer, pt1 db2gse.ST_Point)
```

다음의 INSERT문은 두 행을 삽입합니다. 하나는 Z 좌표나 치수가 없는 점이고, 다른 컬럼에는 Z 좌표와 치수 둘 모두 있습니다.

```
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)))
```

다음의 SELECT문 및 해당 결과 세트에서는 GID 컬럼 및 점의 배정밀도 Z 좌표를 나열합니다. 첫번째 행은 점에 Z 좌표가 없기 때문에 NULL입니다.

```
SELECT gid, db2gse.Z(pt1) "The Z coordinate" FROM Z_TEST

GID         The Z coordinate
----------- ------------------------
          1                        -
          2   +5.00000000000000E+000

  2 record(s) selected.
```

제15장 좌표 시스템

이 장에서는 공간 참조 시스템(SRS)에 관한 참조 정보와 공간 데이터를 해석하는 데 사용된 좌표값에 관한 참조 정보를 제공합니다.

좌표 시스템 개요

공간 참조 시스템의 잘 알려진 텍스트 표현식에서는 공간 참조 시스템 정보에 대한 표준 텍스트 표현식을 제공합니다. 잘 알려진 텍스트 표현식의 정의는 POSC/EPSG 좌표 시스템 데이터 모델 이후에 모델화되었습니다.

공간 참조 시스템이란 지리적 (위도-경도), 프로젝트화된 (X,Y) 또는 지심 (X,Y,Z) 좌표 시스템입니다. 좌표 시스템은 여러 개의 오브젝트로 구성됩니다. 각 오브젝트는 DATUM 또는 UNIT와 같은 대문자 키워드를 가지며 괄호안에 쉼표로 구분하여 정의한 오브젝트 매개변수들이 따라 나옵니다. 일부 오브젝트는 다른 오브젝트들로 구성되므로 결과가 중첩된 구조를 가질 수 있습니다.

주: 구현할 때 대괄호[ ] 대신에 표준 괄호( )를 대체하는 것은 자유이지만 두 괄호 양식 모두 읽을 수 있도록 준비해야 합니다.

대괄호를 사용하는 좌표 시스템의 문자열 표현식에 대한 EBNF(Extended Backus Naur Form) 정의는 다음과 같습니다(괄호 사용에 관한 위의 주를 참조하십시오).

```
<coordinate system> = <projected cs> | <geographic cs> | <geocentric cs>
<projected cs> = PROJCS["<name>", <geographic cs>, <projection>, {<parameter>,}*
                 <linear unit>]
<projection> = PROJECTION["<name>"]
<parameter> = PARAMETER["<name>", <value>]
<value> = <number>
```

데이터 세트의 좌표 시스템은 데이터가 프로젝트화된 좌표인 경우에 PROJCS 키워드로 식별됩니다(기하학 좌표인 경우에는 GEOGCS로, 지심 좌표인 경우에는 GEOCCS로). PROJCS 키워드 뒤에는 프로젝트화된 좌표 시스템을 정의하는 모든 "조각들"이 수반됩니다. 모든 오브젝트의 첫번째 조각은 항상 이름입니다. 프로젝트화된 좌표 시스템 이름 즉, 지리적 좌표 시스템, 맵 공간 방사, 하나 이상의 매개변수 및 선형 치수 뒤에 여러 오브젝트들이 따라 나옵니다. 프로젝트화된 좌표 시스템들은 모두 지리적 좌표 시스템에 근거하므로, 이 절에서는 프로젝트화된 좌표 시스템에 특정적인 조각들을 먼저 설명합니다. 예를 들어, NAD83 기준상의 UTM 존 10N은 다음과 같이 정의됩니다.

```
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]]
```

이름 및 여러 오브젝트들(기준, 자오선 및 각도 치수)이 지리적 좌표 시스템 오브젝트를 차례로 정의합니다.

```
<geographic cs> = GEOGCS["<name>", <datum>, <prime meridian>, <angular unit>]
<datum> = DATUM["<name>", <spheroid>]
<spheroid> = SPHEROID["<name>", <semi-major axis>, <inverse flattening>]
<semi-major axis> = <number>
    (semi-major axis is measured in meters and must be > 0.)
<inverse flattening> = <number>
<prime meridian> = PRIMEM["<name>", <longitude>]
<longitude> = <number>
```

NAD83상의 UTM 존 10에 대한 지리적 좌표 시스템 문자열.

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],
UNIT["Degree",0.0174532925199433]]
```

UNIT 오브젝트는 각도 또는 선형 치수를 나타낼 수 있습니다.

```
<angular unit> = <unit>
<linear unit> = <unit>
<unit> = UNIT["<name>", <conversion factor>]
<conversion factor> = <number>
```

변환 인수는 단위 당 미터수(선형 단위의 경우) 또는 라디안수(각도 단위의 경우)를 지정하며 이는 0보다 커야 합니다.

그러므로 UTM 존 10N의 전체 문자열 표현식은 다음과 같습니다.

```
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]]
```

지심 좌표 시스템은 지리적 좌표 시스템과 유사합니다.

```
<geocentric cs> = GEOCCS["<name>", <datum>, <prime meridian>, <linear unit>]
```

지원되는 선형 단위

표 57. 지원되는 선형 단위

단위	변환 인수
미터	1.0
피트(국제)	0.3048
피트(미국)	12/39.37
수정된 피트(미국)	12.0004584/39.37
Clarke의 피트	12/39.370432
인디안 피트	12/39.370141
링크	7.92/39.370432
링크 (Benoit)	7.92/39.370113

표 57. 지원되는 선형 단위 (계속)

단위	변환 인수
링크(Sears)	7.92/39.370147
측쇄(Benoit)	792/39.370113
측쇄(Sears)	792/39.370147
야드(Indian)	36/39.370141
야드(Sears)	36/39.370147
길(Fathom)	1.8288
항해 마일	1852.0

지원되는 각도 단위

표 58. 지원되는 각도 단위

단위	변환 인수
라디안	1.0
10진수 도	p/180
10진수 분	(p/180)/60
10진수 두 번째	(p/180)/36000
Gon	p/200
Grad	p/200

지원되는 구상체

표 59. 표준 구상체

이름	Semi-major axis	Inverse flattening
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

표 59. 표준 구상체 (계속)

이름	Semi-major axis	Inverse flattening
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
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
Sphere (radius = 1.0)	1	0
Sphere (radius = 6371000 m)	6371000	0

표 59. 표준 구상체 (계속)

이름	Semi-major axis	Inverse flattening
Sphere (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

지원되는 측지학 기준

표 60. 지원되는 측지학 기준

Adindan	Lisbon
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

표 60. 지원되는 측지학 기준 (계속)

Bogota	Nord de Guerre
Bukit Rimpah	NGO 1948
Camacupa	Nord Sahara 1959
Campo Inchauspe	NSWC 9Z-2
Cape	Nouvelle Triangulation Francaise
Carthage	New Zealand Geodetic Datum 1949
Chua	OS (SN) 1980
Conakry 1905	OSGB 1936
Corrego Alegre	OSGB 1970 (SN)
Cote d'Ivoire	Padang 1884
Datum 73	Palestine 1923
Deir ez Zor	Pointe Noire
Deutsche Hauptdreiecksnetz	Provisional South American Datum 1956
Douala	Pulkovo 1942
European Datum 1950	Qatar
European Datum 1987	Qatar 1948
Egypt 1907	Qornoq
유럽 참조 시스템 1989	RT38
Fahud	South American Datum 1969
Gandajika 1970	Sapper Hill 1943
Garoua	Schwarzeck
Geocentric Datum of Australia 1994	Segora
Guyane Francaise	Serindung
Herat North	Stockholm 1938
Hito XVIII 1963	Sudan
Hu Tzu	Shan Tananarive 1925
Hungarian Datum 1972	Timbalai 1948
Indian 1954	TM65
Indian 1975	TM75
Indonesian Datum 1974	Tokyo
Jamaica 1875	Trinidad 1903
Jamaica 1969	Trucial Coast 1948
Kalianpur	Voirol 1875

표 60. 지원되는 측지학 기준 (계속)

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

지원되는 자오선

표 61. 지원되는 자오선

위치	좌표
그리니치	0° 0' 0"
베른	7° 26' 22.5" E
보고타	74° 4' 51.3" W
브뤼셀	4° 22' 4.71" E
페로	17° 40' 0" W
자카르타	106° 48' 27.79" E
리스본	9° 7' 54.862" W
마드리드	3° 41' 16.58" W
파리	2° 20' 14.025" E
로마	12° 27' 8.4" E
스톡홀름	18° 3' 29" E

지원되는 맵 공간 방사

표 62. 지원되는 맵 공간 방사

원통형 공간 방사	의사 원통형 공간 방사
Behrmann	Craster parabolic
Cassini	Eckert I
Cylindrical equal area	Eckert II
Equirectangular	Eckert III

표 62. 지원되는 맵 공간 방사 (계속)

원통형 공간 방사	의사 원통형 공간 방사
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

원추형 공간 방사

표 63. 원추형 공간 방사

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	기타
Polyconic	Alaska series E
Simple conic	Alaska Grid (Modified-Stereographic by Snyder)

방위각형 또는 planar 공간 방사

- 방위각형 등거리
- 일반 수직 근접 측면 투시도
- 해시계 구조원리
- Lambert 방위각형 대등 영역
- 정자법(Orthographic)
- 극성 입체화법(Polar-Stereographic)
- 입체화법

맵 공간 방사 매개변수

표 64. 맵 공간 방사 매개변수

매개변수	설명
central_meridian	x 좌표의 원점으로 선택된 경도선.
scale_factor	일반적으로 맵 공간 방사에서 변형된 부분의 양을 줄이는 데 사용됩니다.
standard_parallel_1	변형 부분이 없는 위도선. "실제 스케일의 위도"에도 사용됩니다.
standard_parallel_2	변형 부분이 없는 위도선.
longitude_of_center	맵 공간 방사의 중앙점을 정의하는 경도.
latitude_of_center	맵 공간 방사의 중앙점을 정의하는 위도.
latitude_of_origin	y 좌표의 원점으로 선택된 위도.
false_easting	x 좌표에 추가됨. 양수값을 제공하는 데 사용됩니다.
false_northing	y 좌표에 추가됨. 양수값을 제공하는 데 사용됩니다.
azimuth	사각 공간 방사의 중앙점을 정의하는 동북쪽 각도.
longitude_of_point_1	맵 공간 방사에 필요한 첫번째 점의 경도.
latitude_of_point_1	맵 공간 방사에 필요한 첫번째 점의 위도.
longitude_of_point_2	맵 공간 방사에 필요한 두 번째 점의 경도.
latitude_of_point_2	맵 공간 방사에 필요한 두 번째 점의 위도.
longitude_of_point_3	맵 공간 방사에 필요한 세 번째 점의 경도.
latitude_of_point_3	맵 공간 방사에 필요한 세 번째 점의 위도.
landsat_number	Landsat 위성의 수.
path_number	특정 위성에 대한 궤도 경로 번호.
perspective_point_height	맵 공간 방사의 원근 화법 점에 대한 지표면에서부터의 높이.
fipszone	State Plane 좌표 시스템 번호.
zone	UTM 존 번호.

제16장 공간 데이터에 대한 파일 형식

이 장에서는 DB2 Spatial Extender의 잘 알려진 표현식에 대해 설명합니다. 표현식은 ESRI에 의해 제공되며 DB2 Spatial Extender에 특정적이지 않기 때문에 잘 알려진으로 서술됩니다. 공간 데이터의 가져오기 및 내보내기를 이해하는 데는 세 가지 종류의 공간값이 중요한 의미를 갖습니다.

- Open GIS Consortium(OGC) 잘 알려진 텍스트 표현식
- OGC 잘 알려진 2진(WKB) 표현식
- ESRI 형상 표현식

OGC 잘 알려진 텍스트 표현식

DB2 Spatial Extender에는 텍스트 설명으로부터 기하학을 생성하는 여러 함수가 있습니다.

ST_GeomFromText
기하학 유형의 텍스트 표현식으로부터 기하학을 작성합니다.

ST_PointFromText
점 텍스트 표현식으로부터 점을 작성합니다.

ST_LineFromText
행스트링 텍스트 표현식으로부터 행스트링을 작성합니다.

ST_PolyFromText
다각형 텍스트 표현식으로부터 다각형을 작성합니다.

ST_MPointFromText
다중점 텍스트 표현식으로부터 다중점을 작성합니다.

ST_MLineFromText
다중 행스트링 텍스트 표현식으로부터 다중 행스트링을 작성합니다.

ST_MPolyFromText
다중 다각형 텍스트 표현식으로부터 다중 다각형을 작성합니다.

텍스트 표현식은 기하학이 ASCII 텍스트 양식으로 교환될 수 있도록 하는 ASCII 텍스트 형식 문자열입니다. 이 함수들은 공간 프로그램 구조를 정의할 필요가 없기 때문에 3세대 또는 4세대 언어(3GL 또는 4GL) 프로그램에 이 함수를 사용할 수 있습니다. ST_AsText 함수는 기존 기하학을 텍스트 표현식으로 변환합니다.

각 기하학 유형에는 잘 알려진 텍스트 표현식이 있는데 이는 유형의 새로운 인스턴스를 구축하고 영숫자 화면용 텍스트 양식으로 기존의 인스턴스를 변환하는 데 모두 사용될 수 있습니다.

기하학의 잘 알려진 텍스트 표현식은 다음과 같이 정의됩니다. 개념 {}*은 괄호 안의 토큰을 0번 이상 반복함을 나타내며, 괄호는 출력 토큰 목록에 나타나지 않습니다.

```
<Geometry Tagged Text> :=
| <Point Tagged Text>
| <LineString Tagged Text>
| <Polygon Tagged Text>
| <MultiPoint Tagged  Text>
| <MultiLineString Tagged Text>
| <MultiPolygon Tagged Text>

<Point Tagged Text> :=
POINT <Point Text>

<LineString Tagged Text> :=
LINESTRING <LineString Text>

<Polygon Tagged Text> :=
POLYGON <Polygon Text>

<MultiPoint Tagged Text> :=
MULTIPOINT <Multipoint Text>

<MultiLineString Tagged Text> :=
MULTILINESTRING <MultiLineString Text>

<MultiPolygon Tagged Text> :=
MULTIPOLYGON <MultiPolygon Text>

<Point Text> := EMPTY
|    <Point>
| Z  <PointZ>
| M  <PointM>
| ZM <PointZM>
```

```
<Point> :=  <x>  <y>
<x> := double precision literal
<y> := double precision literal
<PointZ> :=  <x>  <y>  <z>
<x> := double precision literal
<y> := double precision literal
<z> := double precision literal
<PointM> :=  <x>  <y>  <m>
<x> := double precision literal
<y> := double precision literal
<m> := double precision literal
<PointZM> :=  <x>  <y>  <z>  <m>
<x> := double precision literal
<y> := double precision literal
<z> := double precision literal
<m> := double precision literal

<LineString Text> := EMPTY
|    ( <Point Text >   {,  <Point Text> }*  )
| Z  ( <PointZ Text >  {,  <PointZ Text> }*  )
| M  ( <PointM Text >  {,  <PointM Text> }*  )
| ZM ( <PointZM Text > {,  <PointZM Text> }*  )

<Polygon Text> := EMPTY
| ( <LineString Text > {,< LineString Text > }*  )

<Multipoint Text> := EMPTY
| ( <Point Text >   {,  <Point Text > }*  )

<MultiLineString Text> := EMPTY
| ( <LineString Text > {,< LineString Text>}*  )

<MultiPolygon Text> := EMPTY
| ( < Polygon Text > {,  < Polygon Text > }*  )
```

기본적인 함수 구문은 다음과 같습니다.

```
function (<text description>,<SRID>)
```

공간 참조 식별자인 SRID와 SPATIAL_REFERENCES 테이블에 대한 기본 키가 SPATIAL_REFERENCES 테이블에 저장된 기하학의 공간 참조 시스템을 식별합니다. 공간 컬럼에 기하학을 삽입하기 전에 기하학의 SRID와 공간 컬럼의 SRID가 일치해야 합니다.

텍스트 설명은 작은 따옴표로 묶인 세 가지 기본 구성요소로 구성됩니다. 예를 들면, 다음과 같습니다.

<'*기하학 유형*'> ['*좌표 유형*'] [''*좌표 목록*']

여기서:

기하학 유형

점, 선스트링, 다각형, 다중점, 다중 선스트링 또는 다중 다각형 중 하나입니다.

좌표 유형

기하학에 Z 좌표 또는 치수가 있는지 여부를 지정합니다. 둘 다 기하학에 없는 경우에는 이 인수를 공백으로 남겨 두십시오. 그렇지 않으면 Z 좌표가 들어 있는 기하학의 경우에는 좌표 유형을 Z으로, 치수가 있는 기하학의 경우에는 M으로, 둘 다 있는 기하학의 경우에는 ZM으로 설정하십시오.

좌표 목록

기하학의 정점을 정의합니다. 좌표 목록은 쉼표로 구분되고 괄호로 묶어집니다. 다중 구성요소가 있는 기하학에는 각 구성요소 부분을 묶을 괄호 세트가 필요합니다. 기하학이 비어 있으면 EMPTY 키워드가 좌표를 바꿉니다.

표65에서는 가능한 모든 텍스트 표현식 예에 대한 전체 목록을 보여 줍니다.

표 65. 기하학 유형 및 텍스트 표현식

기하학 유형	텍스트 설명	주석
점	point empty	빈 점
점	point z empty	z 좌표가 있는 빈 점
점	point m empty	치수가 있는 빈 점
점	point zm empty	z 좌표 및 치수가 있는 빈 점
점	point (10.05 10.28)	점
점	point z (10.05 10.28 2.51)	z 좌표가 있는 점
점	point m (10.05 10.28 4.72)	치수가 있는 점
점	point zm (10.05 10.28 2.51 4.72)	z 좌표 및 치수가 있는 점

표 65. 기하학 유형 및 텍스트 표현식 (계속)

기하학 유형	텍스트 설명	주석
선스트링	linestring empty	빈 선스트링
선스트링	linestring z empty	z 좌표가 있는 빈 선스트링
선스트링	linestring m empty	치수가 있는 빈 선스트링
선스트링	linestring zm empty	z 좌표 및 치수가 있는 빈 선스트링
선스트링	linestring (10.05 10.28 , 20.95 20.89)	선스트링
선스트링	linestring z (10.05 10.28 3.09, 20.95 31.98 4.72, 21.98 29.80 3.51)	z 좌표가 있는 선스트링
선스트링	linestring m (10.05 10.28 5.84, 20.95 31.98 9.01, 21.98 29.80 12.84)	치수가 있는 선스트링
선스트링	linestring zm ( )	z 좌표 및 치수가 있는 선스트링
다각형	polygon empty	빈 다각형
다각형	polygon z empty	z 좌표가 있는 빈 다각형
다각형	polygon m empty	치수가 있는 빈 다각형
다각형	polygon zm empty	z 좌표 및 치수가 있는 빈 다각형
다각형	polygon ((10 10, 10 20, 20 20, 20 15, 10 10))	다각형
다각형	polygon z (( ))	z 좌표가 있는 다각형
다각형	polygon m (( ))	치수가 있는 다각형
다각형	polygon zm (( ))	z 좌표 및 치수가 있는 다각형
다중점	multipoint empty	빈 다중점
다중점	multipoint z empty	z 좌표가 있는 빈 다중점
다중점	multipoint m empty	치수가 있는 빈 다중점
다중점	multipoint zm empty	z 좌표 및 치수가 있는 빈 다중점
다중점	multipoint empty	빈 다중점
다중점	multipoint (10 10, 20 20)	두 점이 있는 다중점
다중점	multipoint z (10 10 2, 20 20 3)	z 좌표가 있는 다중점

표 65. 기하학 유형 및 텍스트 표현식 (계속)

기하학 유형	텍스트 설명	주석
다중점	multipoint m (10 10 4, 20 20 5)	치수가 있는 다중점
다중점	multipoint zm (10 10 2 4, 20 20 3 5)	z 좌표 및 치수가 있는 다중점
다중 선스트링	multilinestring empty	빈 다중 선스트링
다중 선스트링	multilinestring z empty	z 좌표가 있는 빈 다중 선스트링
다중 선스트링	multilinestring m empty	치수가 있는 빈 다중 선스트링
다중 선스트링	multilinestring zm empty	z 좌표 및 치수가 있는 빈 다중 선스트링
다중 선스트링	multilinestring (( ))	다중 선스트링
다중 선스트링	multilinestring z (( ))	z 좌표가 있는 다중 선스트링
다중 선스트링	multilinestring m (( ))	치수가 있는 다중 선스트링
다중 선스트링	multilinestring zm (( ))	z 좌표 및 치수가 있는 다중 선스트링
다중 다각형	multipolygon empty	빈 다중 다각형
다중 다각형	multipolygon z empty	z 좌표가 있는 빈 다중 다각형
다중 다각형	multipolygon m empty	치수가 있는 빈 다중 다각형
다중 다각형	multipolygon z	z 좌표 및 치수가 있는 빈 다중 다각형
다중 다각형	multipolygon ((( )))	다중 다각형
다중 다각형	multipolygon z ((( )))	z 좌표가 있는 다중 다각형
다중 다각형	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)))	치수가 있는 다중 다각형
다중 다각형	multipolygon zm ((( )))	z 좌표 및 치수가 있는 다중 다각형

OGC 잘 알려진 2진(WKB) 표현식

DB2 Spatial Extender에는 2진 표현식으로부터 기하학을 생성하는 여러 함수가 있습니다.

ST_GeomFromWKB

기하학 유형의 WKB 표현식으로부터 기하학을 작성합니다.

ST_PointFromWKB

점 WKB 표현식으로부터 점을 작성합니다.

ST_LineFromWKB

선스트링 WKB 표현식으로부터 선스트링을 작성합니다.

ST_PolyFromWKB

다각형 WKB 표현식으로부터 다각형을 작성합니다.

ST_MPointFromWKB

다중점 WKB 표현식으로부터 다중점을 작성합니다.

ST_MLineFromWKB

다중 선스트링 WKB 표현식으로부터 다중 선스트링을 작성합니다.

ST_MPolyFromWKB

다중 다각형 WKB 표현식으로부터 다중 다각형을 작성합니다.

잘 알려진 2진 표현식은 연속적인 바이트 스트림입니다. ODBC 클라이언트와 SQL 데이터베이스 사이에 기하학을 2진 양식으로 교환할 수 있습니다. 이 기하학 함수에는 2진 표현식을 맵핑하는 데 C 프로그래밍 언어 구조의 정의가 필요하기 때문에, 이는 3세대 언어(3GL) 프로그램 내에서 사용될 예정이며, 4세대 언어(4GL) 환경에는 적합하지 않습니다. ST_AsBinary 함수는 기존 기하학값을 잘 알려진 2진 표현식으로 변환합니다.

기하학에 대한 잘 알려진 2진 표현식은 기하학 인스턴스를 숫자 유형 순차로 직렬화하여 얻을 수 있습니다. 이러한 유형들은 세트(부호없는 정수, Double)로부터 그려진 후에, 각 숫자 유형은 바이트 순차로서 직렬화됩니다. 유형은 숫자 유형에 대한 두 가지 잘 알려진, 표준 및 2진 표현식 중 하나를 사용하여 직렬화됩니다(NDR, XDR). 직렬화된 바이트 앞에 오는 1바이트 태그는 기하학 바이트 스트림에 사용되는 특정 2진 암호화(NDR 또는 XDR)를 나타냅니다. 두 기하학의 암호화 사이에 유일한 차이점은 바이트 순서입니다. XDR 암호화는 Big Endian이고 NDR 암호화는 Little Endian입니다.

숫자 유형 정의

부호없는 *정수*(unsigned integer)는 음수가 아닌 정수를 범위 [0, 4294967295]에서 암호화하는 32비트(4 바이트) 데이터 유형입니다.

*Double*은 IEEE 754 배정밀도 형식을 사용하여 배정밀도 숫자를 암호화하는 64비트(8 바이트) 배정밀도 데이터 유형입니다.

이 정의는 XDR 및 NDR 모두에 공통적입니다.

숫자 유형의 XDR(Big Endian) 암호화

부호없는 정수의 XDR 표현식은 Big Endian(최상위 바이트 우선)입니다.

Double의 XDR 표현식은 Big Endian(부호 비트가 첫번째 바이트임)입니다.

숫자 유형의 NDR (Little Endian) 암호화

부호없는 정수의 NDR 표현식은 Little Endian(최상위 바이트 우선)입니다.

Double의 NDR 표현식은 Little Endian(부호 비트가 최종 바이트임)입니다.

NDR 및 XDR간의 변환

부호없는 정수에 대한 NDR 및 XDR 데이터 유형 간의 변환은 간단한 작업입니다. 이에는 바이트 스트림 형태의 각 부호없는 정수 또는 Double 내의 바이트 순서를 역변환하는 작업도 포함됩니다.

WKBGeometry 바이트 스트림의 설명

이 절에서는 기하학에 대한 잘 알려진 2진 표현식에 대해 설명합니다. 기본적인 빌딩 블록은 점에 대한 바이트 스트림이며 이는 두 Double로 구성됩니다. 다른 기하학에 대한 바이트 스트림은 이미 정의된 기하학에 대한 바이트 스트림을 사용하여 빌드됩니다.

```
// Basic Type definitions
// byte : 1 byte
// uint32 : 32 bit unsigned integer  (4 bytes)
// double : double precision number (8 bytes)

// Building Blocks : Point, LinearRing
```

```
Point {
  double x;
  double y;
};
LinearRing   {
  uint32  numPoints;
  Point   points[numPoints];
};
enum wkbGeometryType {
  wkbPoint = 1,
  wkbLineString = 2,
  wkbPolygon = 3,
  wkbMultiPoint = 4,
  wkbMultiLineString = 5,
  wkbMultiPolygon = 6,
};
enum wkbByteOrder {
  wkbXDR = 0,                                    // Big Endian
  wkbNDR = 1                                   // Little Endian
};
WKBPoint {
  byte     byteOrder;
  uint32   wkbType;                                        // 1
  Point    point;
};
WKBLineString {
  byte     byteOrder;
  uint32   wkbType;                                        // 2
  uint32   numPoints;
  Point    points[numPoints];
}

WKBPolygon    {
  byte                byteOrder;
  uint32            wkbType;                               // 3
  uint32            numRings;
  LinearRing        rings[numRings];
}
WKBMultiPoint    {
  byte                byteOrder;
  uint32            wkbType;                               // 4
  uint32            num_wkbPoints;
  WKBPoint            WKBPoints[num_wkbPoints];
}
WKBMultiLineString    {
  byte              byteOrder;
  uint32            wkbType;                               // 5
  uint32            num_wkbLineStrings;
```

```
  WKBLineString     WKBLineStrings[num_wkbLineStrings];
}

wkbMultiPolygon {
  byte              byteOrder;
  uint32            wkbType;                                  // 6
  uint32            num_wkbPolygons;
  WKBPolygon        wkbPolygons[num_wkbPolygons];
}

WKBGeometry  {
  union {
    WKBPoint                point;
    WKBLineString           linestring;
    WKBPolygon              polygon;
    WKBMultiPoint           mpoint;
    WKBMultiLineString      mlinestring;
    WKBMultiPolygon         mpolygon;
  }
};
```

다음 그림에서는 NDR 표현식을 보여 줍니다.

그림 39. NDR 형식의 표현식. 2개의 선형(NR=2)이 있는 유형 다각형(T=3)의 (B=1), 각 링에는 3개의 점(NP=3)가 있음.

WKB 표현식에 대한 주장

기하학에 대한 잘 알려진 2진 표현식은 Geometry Object Model 및 OpenGIS Abstract Specification에 서술된 기하학 유형의 인스턴스를 나타내도록 설계되었습니다.

이러한 주장에는 링, 다각형 및 다중 다각형에 대한 다음과 같은 사항이 포함됩니다.

직선 링

링은 단순하고 닫힌 상태이며 이는 직선 링이 그 자체와 교차할 수 없다는 것을 의미합니다.

다각형 다각형의 경계에 있는 두 개의 선형 링 중 어느 것도 서로 교차할 수 없습니다. 다각형의 경계에 있는 선형 링은 많아야 하나의 점에서 접선(tangent)으로만 교차할 수 있습니다.

다중 다각형

다중 다각형의 요소인 두 다각형의 내부는 교차할 수 없습니다. 다중 다각형의 요소인 두 다각형의 경계는 한정된 숫자의 점에서만 접할 수 있습니다.

ESRI 형상 표현식

DB2 Spatial Extender에는 ESRI 형상 표현식으로부터 기하학을 생성하는 여러 함수가 있습니다. ESRI 형상 표현식은 열린 GIS 잘 알려진 2진 표현식에 의해 지원되는 2차원 표현식 이외에도 선택적으로 Z 좌표와 치수도 지원합니다. 다음 함수들은 ESRI 형상에서 기하학을 생성합니다.

ST_GeometryFromShape

기하학 유형의 형상 표현식으로부터 기하학을 작성합니다.

ST_PointFromShape

점 형상 표현식으로부터 점을 작성합니다.

ST_LineFromShape

행스트링 형상 표현식으로부터 행스트링을 작성합니다.

ST_PolyFromShape

다각형 형상 표현식으로부터 다각형을 작성합니다.

ST_MPointFromShape

다중점 형상 표현식으로부터 다중점을 작성합니다.

ST_MLineFromShape

다중 행스트링 형상 표현식으로부터 다중 행스트링을 작성합니다.

ST_MPolyFromShape

다중 다각형 형상 표현식으로부터 다중 다각형을 작성합니다.

이 함수의 일반적인 구문은 동일합니다. 첫번째 인수는 2진 대형 오브젝트(BLOB) 데이터 유형으로 입력된 형상 표현식입니다. 두 번째 인수는 기하학에 지정하기 위한 공간 참조 식별자 정수입니다. GeometryFromShape 함수의 구문은 다음과 같습니다.

```
db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)
```

이 기하학 함수는 2진 표현식을 맵핑하는데 C 프로그래밍 언어 구조의 정의를 요구하기 때문에, 이는 3GL 프로그램 내에서 사용되며 4GL 환경에서는 적합하지 않습니다. AsShape 함수는 기하학값을 ESRI 형상 표현으로 변환합니다.

형상 유형 0은 형상에 대한 기하학 데이터가 없는 널 형상을 나타냅니다.

값	형상 유형
0	Null Shape
1	Point
3*	PolyLine
5	Polygon
8	MultiPoint
11	PointZ
13	PolyLineZ
15	PolygonZ
18	MultiPointZ
21	PointM
23	PolyLineM
25	PolygonM
28	MultiPointM

주: * 위에 지정되지 않은 형상 유형(2, 4, 6 등)은 나중에 사용하기 위해 예약되어 있습니다.

XY 공간에서의 형상 유형

점

점은 순서 X, Y인 배정밀도 좌표의 쌍으로 구성됩니다.

표 66. 점 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	1	Integer	1	Little
바이트 4	X	X	Double	1	Little
바이트 12	Y	Y	Double	1	Little

다중점

다중점은 점의 콜렉션으로 구성됩니다. 바운딩 상자는 Xmin, Ymin, Xmax, Ymax 순서로 저장됩니다.

표 67. 다중점 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	8	Integer	1	Little
바이트 4	Box	Box	Double	4	Little
바이트 36	NumPoints	NumPoints	Integer	1	Little
바이트 40	Points	Points	Point	NumPoints	Little

PolyLine

PolyLine은 하나 이상의 파트로 구성된 정점의 순서화된 세트입니다. 파트란 둘 이상의 점을 연결한 결과입니다. 점은 서로에게 연결될 수도 연결되지 않을 수도 있습니다. 파트는 서로 교차할 수도 교차하지 않을 수도 있습니다.

이 스펙에서는 동일한 좌표의 연속적인 점을 허용하지 않기 때문에, shapefile 읽기 프로그램에서는 그와 같은 경우를 핸들해야 합니다. 또 한편으로는, 결과로서 생길 수도 있는 변질되거나 0 길이인 파트를 허용하지 않습니다.

PolyLine에 대한 필드는 다음과 같습니다.

Box PolyLine에 대한 바운딩 상자는 Xmin, Ymin, Xmax, Ymax 순서로 저장됩니다.

NumParts

PolyLine에 있는 파트의 수.

NumPoints

모든 파트에 대한 총 점 수.

Parts 길이 NumParts의 배열. 각 PolyLine은 점 배열에 첫번째 점의 색인을 저장합니다. 배열 색인은 0에 대한 것입니다.

Points 길이 NumPoints의 배열. PolyLine의 각 파트에 대한 점이 끝과 끝을 매어 저장됩니다. 파트 2에 대한 점이 파트 1에 대한 점 뒤에 따라 나오는 방식입니다. 파트 배열은 각 파트에 대한 시작 점의 배열 색인을 보유합니다. 점 배열에서는 파트 사이에 분리문자가 없습니다.

표 68. PolyLine 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	3	Integer	1	Little
바이트 4	Box	Box	Double	4	Little
바이트 36	NumParts	NumParts	Integer	1	Little
바이트 40	NumPoints	NumPoints	Integer	1	Little
바이트 44	Parts	Parts	Integer	NumParts	Little
바이트 X	Points	Points	Point	NumPoints	Little

주: X = 44 + 4 * NumParts.

다각형

다각형은 하나 이상의 링으로 구성됩니다. 링은 자체 교차되지 않는 닫혀 있는 루프를 형성하는 네 개 이상 점의 연결된 순서입니다. 다각형에는 여러 개의 외부 링이 포함될 수 있습니다. 링에 대한 정점 또는 방향의 순서는 링의 어느 면이 다각형의 내부인지를 나타냅니다. 정점순으로 링을 따라 걷는 관찰자의 오른쪽에 이웃한 부분이 다각형 내부에의 이웃한 부분입니다. 다각형의 구멍을 정의하는 링의 정점은 시계 반대 방향으로 있습니다. 그러므로, 단일 링으로 된 다각형에 대한 정점들은 항상 시계 방향 순서로 있습니다. 다각형의 링을 파트(part)라고 합니다.

이 스펙에서는 동일한 좌표의 연속적인 점을 허용하지 않기 때문에, shapefile 읽기 프로그램에서는 그와 같은 경우를 핸들해야 합니다. 또 한편으로는, 결과로서 생길 수도 있는 변질되거나 0 길이이거나 0 영역인 파트를 허용하지 않습니다.

다각형에 대한 필드는 다음과 같습니다.

Box Xmin, Ymin, Xmax, Ymax순으로 저장된 다각형에 대한 바운딩 상자.

NumParts

다각형에서의 링 수.

NumPoints

모든 링에 대한 총 점 수.

Parts 길이 NumParts의 배열. 각 링의 경우에 점 배열에 첫번째 점의 색인을 저장합니다. 배열 색인은 0에 대한 것입니다.

Points 길이 NumPoints의 배열. 다각형의 각 링에 대한 점은 끝과 끝을 매어 저장됩니다. 링 2에 대한 점이 링 1에 대한 점 뒤에 따라 나오는 방식입니다. 파트 배열은 각 링에 대한 시작 점의 배열 색인을 보유합니다. 링 사이에는 점 배열에 분리문자가 없습니다.

다각형 형상에 관한 참고사항:

- 링은 닫힌 상태입니다(링의 첫번째 및 마지막 정점은 동일해야 합니다).
- 점 배열에서의 링 순서는 중요하지 않습니다.
- shapefile에 저장된 다각형은 순수해야 합니다. 순수 다각형이란 다음을 의미합니다.
  - 자체 교차되지 않는 다각형. 이는 한 링에 속해 있는 세그먼트가 다른 링에 속해 있는 세그먼트와 교차할 수 없음을 의미합니다. 다각형의 링은 정점에서 서로 접할 수는 있지만 세크먼트를 따라 관통할 수는 없습니다. Colinear 세그먼트는 교차한 것으로 간주합니다.
  - 다각형을 정의하는 선의 ″올바른″ 면에 다각형의 내부가 있는 다각형. 정점순으로 링을 따라 걷는 관찰자의 오른쪽에 이웃한 부분이 다각형 내부입니다. 그러므로, 단일 링으로 된 다각형에 대한 정점들은 항상 시계 방향 순서로 있습니다. 이 다각형에 구멍을 정의하는 링은 시계 반대 방향을 취합니다.

  다각형에 구멍을 정의하는 링이 시계 방향으로 가면 내부가 겹치기 때문에 ″Dirty″ 다각형이 발생합니다.

다각형 인스턴스 예:

v1

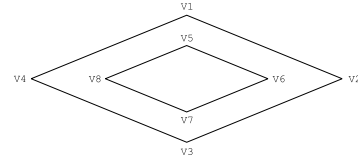


그림 40. 구멍과 8개의 정점이 있는 다각형

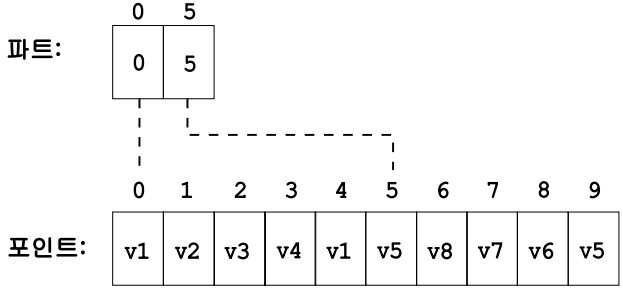


그림 41. 다각형 바이트 스트림의 내용. NumParts는 2와 같고 NumPoints는 10과 같습니다. 도넛(구멍) 다각형에 대한 점 순서가 역전되었음에 유의하십시오.

표 69. 다각형 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	5	Integer	1	Little
바이트 4	Box	Box	Double	4	Little
바이트 36	NumParts	NumParts	Integer	1	Little
바이트 40	NumPoints	NumPoints	Integer	1	Little
바이트 44	Parts	Parts	Integer	NumParts	Little
바이트 X	Points	Points	Point	NumPoints	Little

주: X = 44 + 4 * NumParts.

XY 공간으로 측정한 공간 유형

PointM

PointM은 순서 X, Y인 배정밀도 좌표의 쌍에 치수 M을 더하여 구성됩니다.

표 70. PointM 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	21	Integer	1	Little
바이트 4	X	X	Double	1	Little
바이트 12	Y	Y	Double	1	Little
바이트 20	M	M	Double	1	Little

MultiPointM

MultiPointM에 대한 필드는 다음과 같습니다.

Box Xmin, Ymin, Xmax, Ymax순으로 저장된 MultiPointM에 대한 바운딩 상자.

NumPoints

Points 수.

Points 길이 NumPoints의 Points 배열.

NumMs

뒤에 수반되는 Measures의 수. 이 필드 뒤에 Measures가 수반되지 않으면 NumMs는 두 개의 0 값만 가지거나, Measures가 있으면 NumPoints와 같습니다.

M Range

Mmin, Mmax순으로 저장된 MultiPointM에 대한 최소 및 최대 치수.

M Array

길이 NumPoints의 Measures 배열.

표 71. MultiPointM 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	28	Integer	1	Little
바이트 4	Box	Box	Double	4	Little
바이트 36	NumPoints	NumPoints	Integer	1	Little
바이트 40	Points	Points	Point	NumPoints	Little
바이트 X	NumMs	NumMs	Integer	1	Little
바이트 X+4*	Mmin	Mmin	Double	1	Little
바이트 X+12*	Mmax	Mmax	Double	1	Little
바이트 X+20*	Marray	Marray	Double	NumPoints	Little

주:

1. X = 40 + (16 * NumPoints)
2. * 선택적

PolyLineM

형상 파일 PolyLineM은 하나 이상의 파트로 구성됩니다. 파트란 둘 이상의 점을 연결한 결과입니다. 점은 서로에게 연결될 수도 연결되지 않을 수도 있습니다. 파트는 서로 교차할 수도 교차하지 않을 수도 있습니다.

PolyLineM에 대한 필드는 다음과 같습니다.

Box Xmin, Ymin, Xmax, Ymax순으로 저장된 PolyLineM에 대한 바운딩 상자.

NumParts

PolyLineM에 있는 파트의 수.

NumPoints

모든 파트에 대한 총 점 수.

Parts 길이 NumParts의 배열. 각 파트의 경우에 점 배열에 첫번째 점의 색인을 저장합니다. 배열 색인은 0에 대한 것입니다.

Points 길이 NumPoints의 배열. PolyLineM의 각 파트에 대한 점이 끝과 끝을 매어 저장됩니다. 파트 2에 대한 점이 파트 1에 대한 점 뒤에 따라 나오

는 방식입니다. 파트 배열은 각 파트에 대한 시작 점의 배열 색인을 보유합니다. 점 배열에서는 파트 사이에 분리문자가 없습니다.

NumMs

뒤에 수반되는 Measures의 수. 이 필드 뒤에 Measures가 수반되지 않으면 NumMs는 두 개의 0 값만 가지거나, Measures가 있으면 NumPoints와 같습니다.

M Range

Mmin, Mmax순으로 저장된 PolyLineM에 대한 최소 및 최대 치수.

M Array

길이 NumPoints의 배열. PolyLineM의 각 파트에 대한 치수는 끝과 끝을 매어 저장됩니다. 파트 2에 대한 치수가 파트 1에 대한 치수 뒤에 따라 나오는 방식입니다. 파트 배열은 각 파트에 대한 시작 점의 배열 색인을 보유합니다. 파트 사이에는 치수 배열에 분리문자가 없습니다.

표 72. PolyLineM 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	13	Integer	1	Little
바이트 4	Box	Box	Double	4	Little
바이트 36	NumParts	NumParts	Integer	1	Little
바이트 40	NumPoints	NumPoints	Integer	1	Little
바이트 44	Parts	Parts	Integer	NumParts	Little
바이트 X	Points	Points	Point	NumPoints	Little
바이트 Y	NumMs	NumMs	Integer	1	Little
바이트 Y+4*	Mmin	Mmin	Double	1	Little
바이트 Y+12*	Mmax	Mmax	Double	1	Little
바이트 Y+20*	Marray	Marray	Double	NumPoints	Little

주:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * 선택적

PolygonM

PolygonM은 여러 개의 링으로 구성됩니다. 링은 자체 교차되지 않는 닫혀 있는 루프입니다. 교차는 XYM 공간이 *아닌* XY 공간으로 계산된다는 점에 유의하십시오. PolygonM에는 여러 개의 외부 링이 포함될 수 있습니다. PolygonM의 링을 파트(part)라고 합니다.

PolygonM에 대한 필드는 다음과 같습니다.

Box Xmin, Ymin, Xmax, Ymax순으로 저장된 PolygonM에 대한 바운딩 상자.

NumParts

PolygonM에서의 링수.

NumPoints

모든 링에 대한 총 점 수.

Parts 길이 NumParts의 배열. 각 링의 경우에 점 배열에 첫번째 점의 색인을 저장합니다. 배열 색인은 0에 대한 것입니다.

Points 길이 NumPoints의 배열. PolygonM의 각 링에 대한 점은 끝과 끝을 매어 저장됩니다. 링 2에 대한 점이 링 1에 대한 점 뒤에 따라 나오는 방식입니다. 파트 배열은 각 링에 대한 시작 점의 배열 색인을 보유합니다. 링 사이에는 점 배열에 분리문자가 없습니다.

NumMs

뒤에 수반되는 Measures의 수. 이 필드 뒤에 Measures가 수반되지 않으면 NumMs는 두 개의 0 값만 가지거나 Measures가 있으면 NumPoints와 같습니다.

M Range

Mmin, Mmax순으로 저장된 PolygonM에 대한 최소 및 최대 치수.

M Array

길이 NumPoints의 배열. PolygonM의 각 링에 대한 치수는 끝과 끝을 매어 저장됩니다. 링 2에 대한 치수가 링 1에 대한 치수 뒤에 따라 나오는 방식입니다. 파트 배열은 각 링에 대한 시작 치수의 배열 색인을 보유합니다. 링 사이에는 치수 배열에 분리문자가 없습니다.

PolygonM 형상에 관한 참고사항:

- 링은 닫힌 상태입니다(링의 첫번째 및 마지막 정점은 동일해야 합니다).
- 점 배열에서의 링 순서는 중요하지 않습니다.

표 73. PolygonM 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	15	Integer	1	Little
바이트 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
바이트 40	NumPoints	NumPoints	Integer	1	Little
바이트 44	Parts	Parts	Integer	NumParts	Little
바이트 X	Points	Points	Point	NumPoints	Little
바이트 Y	NumMs	NumMs	Integer	1	Little
바이트 Y+4*	Mmin	Mmin	Double	1	Little
바이트 Y+12*	Mmax	Mmax	Double	1	Little
바이트 Y+20*	Marray	Marray	Double	NumPoints	Little

주:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * 선택적

XYZ 공간에서의 shape types

PointZ

PointZ은 순서 X, Y, Z인 세 배, 두 배 정밀도 좌표의 쌍에 치수 M을 더하여 구성됩니다.

표 74. PointZ 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	11	Integer	1	Little
바이트 4	X	X	Double	1	Little
Byte 12	Y	Y	Double	1	Little
바이트 20	Z	Z	Double	1	Little
바이트 28	Measure	M	Double	1	Little

MultiPointZ

MultiPointZ은 다음과 같은 PointZ 세트를 나타냅니다.

- 바운딩 상자는 Xmin, Ymin, Xmax, Ymax 순서로 저장됩니다.
- 바운딩 Z 범위는 Zmin, Zmax 순서로 저장됩니다. 바운딩 M 범위는 Mmin, Mmax 순서로 저장됩니다.

표 75. MultiPointZ 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	18	Integer	1	Little
바이트 4	Box	Box	Double	4	Little
바이트 36	NumPoints	NumPoints	Integer	1	Little
바이트 40	Points	Points	Point	NumPoints	Little
바이트 X	Zmin	Zmin	Double	1	Little
바이트 X+8	Zmax	Zmax	Double	1	Little
바이트 X+16	Zarray	Zarray	Double	NumPoints	Little
바이트 Y	NumMs	NumMs	Integer	1	Little
바이트 Y+4*	Mmin	Mmin	Double	1	Little
바이트 Y+12*	Mmax	Mmax	Double	1	Little
바이트 Y+20*	Marray	Marray	Double	NumPoints	Little

주:

1. X = 40 + (16 * NumPoints); Y = X + 16 + (8 * NumPoints)
2. * 선택적

PolyLineZ

PolyLineZ은 하나 이상의 파트로 구성됩니다. 파트란 둘 이상의 점을 연결한 결과입니다. 점은 서로에게 연결될 수도 연결되지 않을 수도 있습니다. 파트는 서로 교차할 수도 교차하지 않을 수도 있습니다.

PolyLineZ에 대한 필드는 다음과 같습니다.

Box Xmin, Ymin, Xmax, Ymax순으로 저장된 PolyLineZ에 대한 바운딩 상자.

NumParts

PolyLineZ에 있는 파트의 수.

NumPoints

모든 파트에 대한 총 점 수.

Parts 길이 NumParts의 배열. 각 파트의 경우에 점 배열에 첫번째 점의 색인을 저장합니다. 배열 색인은 0에 대한 것입니다.

Points 길이 NumPoints의 배열. PolyLineZ의 각 파트에 대한 점이 끝과 끝을 매어 저장됩니다. 파트 2에 대한 점이 파트 1에 대한 점 뒤에 따라 나오는 방식입니다. 파트 배열은 각 파트에 대한 시작 점의 배열 색인을 보유합니다. 점 배열에서는 파트 사이에 분리문자가 없습니다.

Z Range

Zmin, Zmax순으로 저장된 PolyLineZ에 대한 최소 및 최대 Z 값.

Z Array

길이 NumPoints의 배열. PolyLineZ의 각 파트에 대한 Z 값이 끝과 끝을 매어 저장됩니다. 파트 2에 대한 Z 값이 파트 1에 대한 Z 값 뒤에 따라 나오는 방식입니다. 파트 배열은 각 파트에 대한 시작 점의 배열 색인을 보유합니다. 파트 사이에는 Z 배열에 분리문자가 없습니다.

NumMs

뒤에 수반되는 Measures의 수. 이 필드 뒤에 Measures가 수반되지 않으면 NumMs는 두 개의 0 값만 가지거나, Measures가 있으면 NumPoints와 같습니다.

M Range

Mmin, Mmax순으로 저장된 PolyLineZ에 대한 최소 및 최대 치수.

M Array

길이 NumPoints의 배열. PolyLineZ의 각 파트에 대한 치수가 끝과 끝을 매어 저장됩니다. 파트 2에 대한 치수가 파트 1에 대한 치수 뒤에 따라 나오는 방식입니다. 파트 배열은 각 파트에 대한 시작 치수의 배열 색인을 보유합니다. 파트 사이에는 치수 배열에 분리문자가 없습니다.

표 76. PolyLineZ 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	13	Integer	1	Little
바이트 4	Box	Box	Double	4	Little

표 76. PolyLineZ 바이트 스트림 내용 (계속)

위치	필드	값	유형	번호	순서
바이트 36	NumParts	NumParts	Integer	1	Little
바이트 40	NumPoints	NumPoints	Integer	1	Little
바이트 44	Parts	Parts	Integer	NumParts	Little
바이트 X	Points	Points	Point	NumPoints	Little
바이트 Y	Zmin	Zmin	Double	1	Little
바이트 Y+8	Zmax	Zmax	Double	1	Little
바이트 Y+16	Zarray	Zarray	Double	NumPoints	Little
바이트 Z	NumMs	NumMs	Integer	1	Little
바이트 Z+4*	Mmin	Mmin	Double	1	Little
바이트 Z+12*	Mmax	Mmax	Double	1	Little
바이트 Z+20*	Marray	Marray	Double	NumPoints	Little

주:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints), Z = Y + 16 + (8 * NumPoints)
2. * 선택적

PolygonZ

PolygonZ은 여러 개의 링으로 구성됩니다. 링은 자체 교차되지 않는 닫혀 있는 루프입니다. PolygonZ에는 여러 개의 외부 링이 포함될 수 있습니다. PolygonZ의 링을 파트(part)라고 합니다.

PolygonZ에 대한 필드는 다음과 같습니다.

Box Xmin, Ymin, Xmax, Ymax순으로 저장된 PolygonZ에 대한 바운딩 상자.

NumParts

PolygonZ에서의 링수.

NumPoints

모든 링에 대한 총 점 수.

Parts 길이 NumParts의 배열. 각 링의 경우에 점 배열에 첫번째 점의 색인을 저장합니다. 배열 색인은 0에 대한 것입니다.

Points 길이 NumPoints의 배열. PolygonZ의 각 링에 대한 점은 끝과 끝을 매어 저장됩니다. 링 2에 대한 점이 링 1에 대한 점 뒤에 따라 나오는 방식입니다. 파트 배열은 각 링에 대한 시작 점의 배열 색인을 보유합니다. 링 사이에는 점 배열에 분리문자가 없습니다.

Z Range

Zmin, Zmax순으로 저장된 arc에 대한 최소 및 최대 Z 값.

Z Array

길이 NumPoints의 배열. PolygonZ의 각 링에 대한 Z 값은 끝과 끝을 매어 저장됩니다. 링 2에 대한 Z 값이 링 1에 대한 Z 값 뒤에 따라 나오는 방식입니다. 파트 배열은 각 링에 대한 시작 Z 값의 배열 색인을 보유합니다. Z 값 배열에서는 링 사이에 분리문자가 없습니다.

NumMs

뒤에 수반되는 Measures의 수. 이 필드 뒤에 Measures가 수반되지 않으면 NumMs는 두 개의 0 값만 가지거나, Measures가 있으면 NumPoints와 같습니다.

M Range

Mmin, Mmax순으로 저장된 PolygonZ에 대한 최소 및 최대 치수.

M Array

길이 NumPoints의 배열. PolygonZ의 각 링에 대한 치수는 끝과 끝을 매어 저장됩니다. 링 2에 대한 치수가 링 1에 대한 치수 뒤에 따라 나오는 방식입니다. 파트 배열은 각 링에 대한 시작 치수의 배열 색인을 보유합니다. 치수 배열에서는 링 사이에 분리문자가 없습니다.

PolygonZ 형상에 관한 참고사항:

- 링은 닫힌 상태입니다(링의 첫번째 및 마지막 정점은 동일해야 합니다).
- 점 배열에서의 링 순서는 중요하지 않습니다.

표 77. PolygonZ 바이트 스트림 내용

위치	필드	값	유형	번호	순서
바이트 0	Shape Type	15	Integer	1	Little
바이트 4	Box	Box	Double	4	Little
바이트 36	NumParts	NumParts	Integer	1	Little

표 77. PolygonZ 바이트 스트림 내용 (계속)

위치	필드	값	유형	번호	순서
바이트 40	NumPoints	NumPoints	Integer	1	Little
바이트 44	Parts	Parts	Integer	NumParts	Little
바이트 X	Points	Points	Point	NumPoints	Little
바이트 Y	Zmin	Zmin	Double	1	Little
바이트 Y+8	Zmax	Zmax	Double	1	Little
바이트 Y+16	Zarray	Zarray	Double	NumPoints	Little
바이트 Z	NumMs	NumMs	Integer	1	Little
바이트 Z+4*	Mmin	Mmin	Double	1	Little
바이트 Z+12*	Mmax	Mmax	Double	1	Little
바이트 Z+20*	Marray	Marray	Double	NumPoints	Little

제3부 부록 및 끝머리

주의사항

다른 국가에서는 이 책에 설명된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 사용자 영역에서 현재 사용 가능한 제품 및 서비스에 관한 정보는 해당 지역 IBM 대표부에 문의하십시오. IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 반드시 IBM 제품, 프로그램 또는 서비스만을 사용하라는 것을 의미하는 것은 아닙니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수 있습니다. 그러나 모든 비-IBM 제품, 프로그램 또는 서비스의 조작을 평가하고 검증하는 것은 사용자의 책임입니다.

IBM은 이 책의 주요 주제에 대하여 특허권을 소유하고 있거나 특허권 적용 보류 중에 있습니다. 이 책의 제공으로 이러한 특허권에 대한 사용권을 부여하는 것은 아닙니다. 사용권 조회를 다음의 주소로 서면으로 전송할 수 있습니다.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
지적 재산권부

2바이트(DBCS) 정보에 관한 사용권 조회는 해당 국가의 IBM 지적 재산권부에 문의하거나 다음의 주소에 서면으로 조회를 전송하십시오.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 사항은 그러한 제공이 지역의 법규와 일치 하지 않는 영국 또는 모든 다른 국가에는 적용되지 않습니다. IBM 주식회사는 이 책을 비-침해성, 상품성 또는 특정 목적에 대한 적합성에 대한 명시적 또는 암시적 보증을 포함하여, 어떠한 종류의 보증도 하지 않고『있는 그대로』제공합니다. 일부 국가에서는 특정 트랜잭션에 대한 명시적 또는 암시적 보증의 거부를 허용하지 않으므로, 이 내용이 사용자에게 적용되지 않을 수도 있습니다.

이 책은 기술적 부정확성이나 철자상의 오류를 포함할 수 있습니다. 이 책에 포함된 정보는 주기적으로 변경되며, 이러한 변경사항은 개정판에 적용됩니다. IBM은 언제든지 아무런 통지없이 이 책에 설명된 제품(들) 및/또는 프로그램(들)을 개선 및/또는 변경할 수 있습니다.

이 책에서 타사의 웹 사이트를 언급한 것은 단지 편의를 위해서일 뿐이며 이런 웹 사이트를 추천하려는 의도는 아닙니다. 이런 웹 사이트의 데이터가 이 IBM 제품에 대한 데이터의 일부는 아니므로 이런 웹 사이트 사용에 대한 책임은 사용자가 져야 합니다.

IBM은 독자가 제공한 정보를 적절한 방식으로 사용하거나 배포할 수 있으며, 제공한 독자는 이에 대해 책임을 지지 않습니다.

이 프로그램의 사용권을 보유한 고객이 (i) 별도로 작성된 프로그램과 기타 프로그램(이 프로그램 포함) 사이의 정보 교환 및 (ii) 교환된 정보의 공동 사용을 가능케할 목적으로 이 프로그램에 관한 정보를 필요로 하는 사용권자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
소프트웨어 사업본부

조항 및 조건(일부의 경우, 요금 지불)이 적합하면, 이러한 정보를 얻을 수 있습니다.

이 책에 설명된 사용권이 부여된 프로그램과 여기에 사용할 수 있는 모든 사용권이 부여된 데이터는 IBM 고객 협약, IBM 국제 프로그램 사용권 협약 또는 동등한 쌍방간의 협약의 조건에 따라 IBM에서 제공합니다.

여기에 제시된 어떠한 성능 데이터는 주위 환경에 따라 결정될 수 있습니다. 따라서, 다른 운영 체제에서 제시된 결과 값과 다를 수 있습니다. 몇몇 측정값은 개발 단계에서 얻은 값일 수 있습니다. 따라서 일반적인 사용자 시스템에서 얻은 값과 다를 수 있습니다. 또한 몇몇 측정값은 보외법을 통해 측정된 값입니다. 실제 값과는 다를 수 있습니다. 이 책의 사용자는 사용자의 특정 환경에 맞게 적용가능한 데이터를 변경해야 합니다.

타사 제품에 대한 정보는 이들 제품 공급업체, 발표 및 기타 공용으로 사용할 수 있는 소스에서 구한 것입니다. IBM은 이들 제품을 테스트하지 않았으므로, 타사 제품에 관련된 성능 정확도, 호환성 또는 기타 청구에 대해 확정할 수 없습니다. 타사 제품에 대한 성능상의 질문은 이들 제품 공급업체에게 보내셔야 합니다.

IBM의 앞으로의 방향 또는 의도에 관한 모든 언급은 변경되거나 아무런 통지없이 철회될 수 있으며, 이는 단지 목적 또는 목표만을 나타내는 것입니다.

이 정보는 일상적인 비지니스 처리에 사용되는 데이터와 보고서의 예가 들어 있을 수 있습니다. 이를 가능한 완벽하게 설명하기 위해, 예제에 개인, 회사, 상표 및 제품의 이름이 포함됩니다. 이러한 이름은 모두 허구이며 실제 비지니스 조직에서 사용되는 이름 및 주소와의 유사성은 전적으로 우연에 의한 것입니다.

저작원 사용권:

이 정보에는 여러 운영 체제에서 프로그래밍 소스 언어로 예제 응용프로그램이 들어 있을 수 있습니다. 샘플 프로그램이 작성된 운영 플랫폼에 대한 응용프로그래밍 인터페이스를 준수하는 응용프로그램을 개발, 사용, 마켓팅 또는 분배하기 위해 이들 샘플 프로그램을 IBM에 댓가를 지불하지 않고 어떠한 형태로든 복사, 수정 및 분배할 수 있습니다. 이들 예제는 모든 조건에서 완전히 테스트되지 않았습니다. 따라서 IBM은 이들 프로그램의 신뢰성, 서비스 가능성 및 작동을 보장하거나 암시할 수 없습니다.

이들 예제 프로그램의 각각의 복사본이나 특정 부분은 다음과 같은 사용권 주의사항을 포함해야 합니다.

© (화사명) (연도). 이 코드의 일부는 IBM사 샘플 프로그램으로부터 비롯되었습니다. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

등록상표

별표(*)로 표시된 다음의 용어는 전세계에서 IBM의 상표입니다.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extender
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

다음 용어는 해당 회사의 상표 또는 등록 상표입니다.

Microsoft, Windows 및 Windows NT는 Microsoft Corporation의 상표 또는 등록 상표입니다.

Java 또는 모든 Java 관련 상표 및 로고 그리고 Solaris는 전세계에서 Sun Microsystems, Inc.의 상표입니다.

Tivoli 및 NetView는 전세계에서 Tivoli Systems Inc.의 상표입니다.

UNIX는 전세계에서 X/Open Company Limited가 독점권을 갖는 등록 상표입니다.

두 개의 별표(**)가 붙은 기타 회사 이름, 제품 이름 또는 서비스 이름은 해당 회사의 상표이거나 서비스 표시입니다.

색인

[가]

[나]

[다]

[마]

[바]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

A

B

D

E

G

I

J

L

M

N

P

S

U

W

X

Y

Z

IBM 문의

기술적인 문제가 발생한 경우에는 DB2 고객 지원 센터에 문의하기 전에 *문제점 해결 안내서*에서 제안한 조치를 검토하고 실행해 보십시오. 이것은 DB2 고객 지원 부서로 하여금 사용자를 보다 더 잘 지원할 수 있도록 사용자가 모을 수 있는 정보를 제공합니다.

DB2 Universal Database 제품에 대한 정보나 주문은 그 지역의 IBM 영업 담당자나 공인 IBM 소프트웨어 재판매업자에게 문의하십시오.

미국내에 거주하는 경우는 다음 번호중 하나로 전화할 수 있습니다.

- 고객 지원을 받으려면, 1-800-237-5511.
- 사용가능한 서비스 옵션을 알려면, 1-888-426-4343.

제품 정보

미국내에 거주하는 경우는 다음 번호중 하나로 전화할 수 있습니다.

- 제품을 주문하거나 일반 정보를 알려면 1-800-IBM-CALL (1-800-426-2255) 또는 1-800-3IBM-OS2 (1-800-342-6672).
- 서적을 주문하려면 1-800-879-2755.

http://www.ibm.com/software/data/

DB2 WWW 페이지에서는 뉴스, 제품 설명, 교육 일정 등등에 대한 현재 DB2 정보를 제공합니다.

http://www.ibm.com/software/data/db2/library/

DB2 제품 및 서비스 기술 라이브러리에서는 자주 질문하는 물음, 수정내용, 서적 및 최신 DB2 기술 정보에 액세스할 수 있습니다.

주: 이 정보는 영어로만 되어 있습니다.

http://www.elink.ibmlink.ibm.com/pbl/pbl/

여기에서는 책을 웹 사이트에서 주문할 수 있는 방법을 제공합니다.

http://www.ibm.com/education/certify/

IBM 웹 사이트에서 기술 전문 인증 프로그램은 DB2를 포함하여 다른 IBM 제품의 기술 전문 인증 테스트 정보를 제공합니다.

ftp.software.ibm.com

Anonymous로 로그온하십시오. `/ps/products/db2` 디렉토리에서, DB2와 많은 관련 제품에 관한 데이터, 수정사항, 도구 등을 찾을 수 있습니다.

comp.databases.ibm-db2, bit.listserv.db2-l

이러한 인터넷 뉴스 그룹으로 사용자는 DB2 제품에 대한 자신의 사용 경험을 토론할 수 있습니다.

Compuserve에서, GO IBMDB2

이 명령을 입력하여 IBM DB2 계열 포럼을 액세스하십시오. 모든 DB2 제품은 이들 포럼을 통해 지원됩니다.

미국 외에 있는 IBM에 연락하는 방법에 대해서는 *IBM 소프트웨어 지원 핸드북*의 부록 A를 참조하십시오. 이 문서에 액세스하려면, 웹 사이트 http://www.ibm.com/support/로 가서 페이지 맨 밑에 있는 IBM Software Support Handbook 링크를 클릭하십시오.

주: 일부 국가에서는 IBM-인증 딜러가 IBM 지원 센터 대신 이들 딜러의 지원 조직에 연락해야 합니다.

IBM 한글 지원에 관한 설문

FAX : (02) 781-7778

보내 주시는 의견은 더 나은 고객 지원 체제를 위한 귀중한 자료가 됩니다.
독자 여러분의 좋은 의견을 기다립니다.

책 제목: IBM® DB2® Spatial Extender
사용자 안내 및 참조서
버전 7

책 번호: SA30-1045-00

성 명		직위/담당업무	
회 사 명		부 서 명	
주 소			
전화번호		팩스번호	
전자우편 주소			
사용중인 시스템	o 중대형 서버 o UNIX 서버 o PC 및 PC 서버		

1. IBM에서 제공하는 한글 책자와 영문 책자 중 어느 것을 더 좋아하십니까? 그 이유는 무엇입니까?
   ☐ 한글 책자 ☐ 영문 책자
   (이유:)

2. 본 책자와 해당 소프트웨어에서 사용된 한글 용어에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

3. 본 책자와 해당 소프트웨어에서 번역 품질에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

4. 본 책자의 인쇄 상태에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

5. 한글 소프트웨어 및 책자가 지원되는 분야에 대해 귀하는 어떻게 생각하십니까?
   ☐ 한글 책자를 늘려야 함 ☐ 현재 수준으로 만족
   ☐ 그다지 필요성을 느끼지 않음

6. IBM은 인쇄물 형식(hardcopy)과 화면 형식(softcopy)의 두 종류로 책자를 제공합니다. 어느 형식을 더 좋아하십니까?
   ☐ 인쇄물 형식(hardcopy) ☐ 화면 형식(softcopy) ☐ 둘 다

IBM 한글 지원 서비스에 대해 기타 제안사항이 있으시면 적어주십시오.

_____

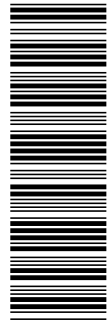
_____

_____

☺ 설문에 답해 주셔서 감사합니다.
귀하의 의견은 저희에게 매우 소중한 것이며, 고객 여러분들께 보다 좋은 제품을 제공해 드리기 위해 최선을 다하겠습니다.

부품 번호: CT7C0KO

Printed in Singapore



CT7C0KO

SA30-1045-00

Spine information:

IBM® DB2® Spatial Extender

DB2 Spatial Extender 사용자 안내 및 참조서

버전 7