IBM® DB2® Universal Database

SQL 참조서

버전 7

SA30-0997-00, SA30-0998-00

IBM® DB2® Universal Database

SQL 참조서

버전 7

SA30-0997-00, SA30-0998-00

이 책의 정보와 지원하는 제품을 사용하기 전에 반드시 1621 페이지의 『부록S. 주의사항』을 읽으십시오.

목차

제1장 소개

이 장에서는

- 이 책의 목적과 사용자를 알려줍니다.
- 이 책의 구성과 사용법에 대해 설명합니다.
- 구문 도표 표기법, 매뉴얼 전체에 사용된 명명 규칙과 강조표시 규칙에 대해 설명합니다.
- 관련 문서를 나열합니다.
- 제품군의 개요를 나타냅니다.

이 책의 사용자

이 책은 SQL을 사용하여 데이터베이스를 액세스하려는 사용자를 위한 것입니다. 주로 프로그래머와 데이터베이스 관리자를 위한 것이지만 명령행 처리기를 사용하는 일반 사용자도 사용할 수 있습니다.

이 책은 학습서가 아닌 참조서입니다. 여기에서는 사용자가 응용프로그램을 작성하므로, 데이터베이스 관리 프로그램의 모든 함수가 있다고 가정합니다.

이 책의 사용법

이 책에서는 DB2 Universal Database 버전 7에 사용된 SQL 언어를 정의합니다. 관계형 데이터베이스 개념, 언어 요소, 함수, 조회 양식 및 SQL문의 구문과 의미에 대한 정보를 원할 경우 이 책을 참조서로 사용하십시오. 부록에서는 중요한 구성요소에 대한 정보와 제한사항을 찾아볼 수 있습니다.

이 책의 구성

이 참조서는 두 볼륨으로 나눠집니다. 볼륨 1에는 다음과 같은 내용이 수록되어 있습니다.

- 『제1장 소개』에는, 이 책의 목적, 대상 및 사용에 대해 설명합니다.

- 9 페이지의 『제2장 개념』에는, 관계형 데이터베이스와 SQL의 기본적인 개념에 대해 설명합니다.
- 73 페이지의 『제3장 언어 요소』에는, 많은 SQL문에 공통인 SQL과 언어 요소의 기본 구문에 대해 설명합니다.
- 239 페이지의 『제4장 함수』에는, SQL 컬럼 및 스칼라 함수의 구문 도표, 의미 설명, 규칙 및 사용 예가 나와 있습니다.
- 433 페이지의 『제5장 조회』는 조회의 여러 가지 양식에 대해 설명합니다.
- 볼륨 1에 포함된 부록에는 다음과 같은 정보가 있습니다.
  - 1253 페이지의 『부록A. SQL 한계』에는 SQL 제한사항이 나와 있습니다.
  - 1261 페이지의 『부록B. SQL 통신(SQLCA)』에는 SQLCA 구조가 나와 있습니다.
  - 1267 페이지의 『부록C. SQL 설명자 영역(SQLDA)』에는 SQLDA 구조가 나와 있습니다.
  - 1281 페이지의 『부록D. 카탈로그 뷰』에는 데이터베이스에 대한 카탈로그 뷰가 나와 있습니다.
  - 1389 페이지의 『부록E. 구조화 유형 사용을 위한 카탈로그 뷰』에는 데이터베이스에 대한 구조화 유형 카탈로그 뷰가 나와 있습니다.
  - 1403 페이지의 『부록F. 연합 시스템』에는 연합 시스템에 대한 옵션과 유형 맵핑이 나와 있습니다.
  - 1419 페이지의 『부록G. 샘플 데이터베이스 테이블』에는 예제에 사용된 샘플 테이블이 나와 있습니다.
  - 1443 페이지의 『부록H. 예약된 스키마 이름 및 예약어』에는 예약된 스키마 이름과 IBM SQL 및 ISO/ANS SQL92 표준 예약어가 나와 있습니다.
  - 1449 페이지의 『부록I. 분리 레벨 비교』에는 분리 레벨의 요약이 나와 있습니다.
  - 1451 페이지의 『부록J. 트리거와 제한조건의 상호 작용』에서는 트리거와 참조 제한조건의 상호작용에 대해 설명합니다.
  - 1455 페이지의 『부록K. Explain 테이블 및 정의』에는 Explain 테이블과 테이블이 정의된 방법이 나와 있습니다.

- 1489 페이지의 『부록L. Explain 레지스터 값』에는 CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터 값 서로간의 상호 작용과 이들 값과 PREP 및 BIND 명령과의 상호 작용에 대해 설명되어 있습니다.
- 1493 페이지의 『부록M. 순환 예: 부품표(BOM: Bill of Materials)』에는 순환 조회의 예가 나와 있습니다.
- 1499 페이지의 『부록N. 예외 테이블』에는 SET INTEGRITY문과 함께 사용되는 사용자 작성 테이블에 대한 정보가 있습니다.
- 1505 페이지의 『부록O. 일본어 및 대만어의 EUC 고려사항』에는 EUC 문자 집합 사용시의 고려사항이 나열되어 있습니다.
- 1515 페이지의 『부록P. DATALINK를 위한 BNF 스펙』에는 DATALINK에 대한 BNF 스펙이 나와 있습니다.

볼륨 2에는 다음과 같은 내용이 수록되어 있습니다.

- 497 페이지의 『제6장 SQL문』에는, 모든 SQL문의 구문 도표, 의미 설명, 규칙 및 그 예가 나와 있습니다.
- 1207 페이지의 『제7장 SQL 프로시듀어』에는, 구문 도표, 의미적 설명, 규칙 및 SQL 프로시듀어문의 예가 나와 있습니다.

구문 도표 읽는법

이 책 전반에 걸쳐, 다음과 같이 정의된 구조를 사용하여 구문을 설명합니다.

행의 경로를 따라 왼쪽에서 오른쪽, 위에서 아래로 구문 도표를 읽습니다.

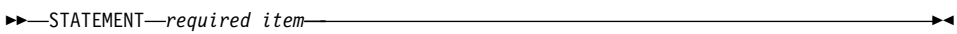
▶▶── 기호는 명령문의 시작을 나타냅니다.

──▶ 기호는 명령문 구문이 다음 행에서 계속됨을 나타냅니다.

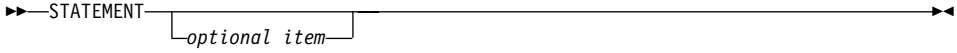
▶── 기호는 명령문이 이전 행으로부터 계속됨을 나타냅니다.

──▶◀ 기호는 명령문의 끝을 나타냅니다.

필수 항목은 수평 행(main path)에 나타납니다.



선택 항목은 주 경로 아래에 나타납니다.

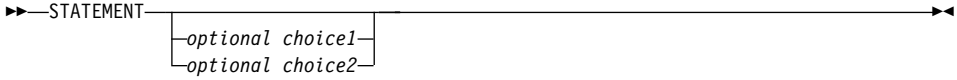


선택 항목이 기본 경로에 나타나면, 그 항목은 명령문 실행에 영향을 주지 않고 읽기 전용으로만 사용됩니다.

둘 이상의 항목에서 선택하는 경우, 스택으로 표시됩니다.

항목 중 하나를 반드시 선택*해야*하는 경우, 스택 중 한 항목이 주 경로에 나타납니다.

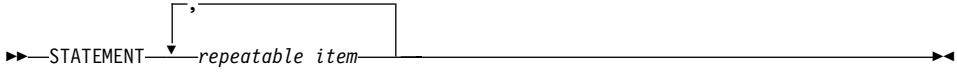
아무 항목도 선택하지 않는 것도 한 옵션인 경우, 전체 스택이 주 경로의 아래에 나타납니다.



항목 중 하나가 기본값인 경우, 주 경로의 위에 나타나고, 나머지 선택사항은 아래에 나타납니다.

주 경로 위에서 왼쪽으로 돌아오는 화살표는 반복될 수 있는 항목을 나타냅니다. 이런 경우, 반복되는 항목은 하나 이상의 공백으로 분리되어야 합니다.

반복 화살표에 쉼표가 있는 경우, 쉼표로 반복 항목을 분리해야 합니다.



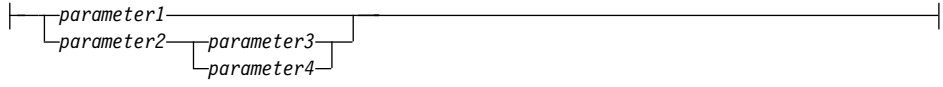
스택 위의 반복 화살표는 스택된 항목 중에서 둘 이상을 선택할 수 있거나 하나의 선택을 반복할 수 있음을 나타냅니다.

키워드는 대문자로 표시됩니다(예를 들면, FROM). 이들은 보이는 대로 정확히 입력해야 합니다. 변수는 소문자 또는 한글로 표시됩니다(예, 컬럼 이름, column-name). 이들은 구문에서 사용자가 제공하는 이름이나 값을 나타냅니다.

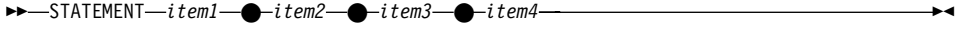
구두점, 괄호, 산술 연산자 또는 그 외의 기호들은 구문의 일부로 입력해야 합니다.

때때로 하나의 변수가 여러 매개변수의 집합을 나타내기도 합니다. 예를 들어, 다음 도표에서, 변수 parameter-block은 헤드가 **parameter-block**인 도표의 해석에 의해 대체됩니다.

매개변수 블록:



『굵은 점』(●) 사이에 오는 인접 세그먼트는 순서에 관계없이 지정할 수 있습니다.



위 도표는 항목2와 항목3을 임의의 순서로 지정할 수 있음을 보여줍니다. 다음의 두 가지가 모두 유효합니다.

```
STATEMENT 항목1 항목2 항목3 항목4
STATEMENT 항목1 항목3 항목2 항목4
```

이 책에서 사용된 규약

이 절에서는 이 매뉴얼에서 일관되게 사용된 규약을 지정합니다.

오류 조건

오류 조건은 괄호 안에 오류와 연관된 SQLSTATE를 나열하여 매뉴얼의 텍스트에 표시됩니다. 예를 들어, 시그니처가 중복되면 SQL 오류(SQLSTATE 42723)가 발생합니다.

강조표시 규칙

다음 규약이 이 책에서 사용됩니다.

굵은체	명령, 키워드 및 시스템이 그 이름을 사전정의한 그외 항목을 나타냅니다.
이탤릭체	다음 중 하나를 나타냅니다. • 사용자가 제공해야 하는 이름 또는 변수 • 일반적인 강조 • 새로운 용어 소개 • 또 다른 정보 소스에 대한 언급
가는체	다음 중 하나를 나타냅니다. • 파일 및 디렉토리 • 사용자가 명령 프롬프트나 창에 입력해야 하는 정보 • 특정 데이터값의 예 • 시스템이 표시하는 것과 유사한 예 • 시스템 메시지의 예

이 책의 관련 책

다음 책은 응용프로그램을 준비하는 데 유용합니다.

- *관리 안내서*

  - 지역적으로 또는 클라이언트/서버 환경에서 액세스되는 데이터베이스의 설계, 구현 및 유지보수에 필요한 정보가 들어 있습니다.
- *응용프로그램 개발 안내서*
  - Embedded SQL과 API를 사용하여 데이터베이스를 액세스하는 응용프로그램을 코딩하고, 컴파일 및 수행하는 방법과 응용프로그램 개발 프로세스에 대해 설명합니다.
- *Spatial Extender 사용자 안내 및 참조서*
  - 응용프로그램을 작성하여 그래픽 정보 시스템(GIS)을 작성하고 사용하는 방법에 대해 설명합니다. GIS 작성 및 사용에는 자원이 있는 데이터베이스를 제공한 후, 위치, 거리 및 영역 내의 분산과 같은 정보를 확보하기 위해 데이터를 조회하는 작업이 포함됩니다.
- *IBM SQL 참조서*
  - 이 매뉴얼에는 IBM의 데이터베이스 제품 라이브러리에서 공통적인 SQL의 모든 요소가 들어 있습니다. 이것은 IBM 데이터베이스를 사용하여 이식가능한 프로그램을 준비하는 데 도움이 되는 제한사항과 규칙을 제공합니다. 또한, 다음 표준과 SQL92E, XPG4-SQL, IBM-SQL 및 IBM 관계형 데이터베이스 제품들 사이의 SQL 확장과 비호환성에 대한 목록을 제공합니다.
- *American National Standard X3.135-1992, Database Language SQL*
  - SQL의 ANSI 표준 정의가 들어 있습니다.
- *ISO/IEC 9075:1992, Database Language SQL*
  - SQL의 1992 ISO 표준 정의가 들어 있습니다.
- *ISO/IEC 9075-2:1999, Database Language SQL -- Part 2: Foundation (SQL/Foundation)*
  - SQL의 1999 ISO 표준 정의에서 많은 부분을 포함하고 있습니다.
- *ISO/IEC 9075-4:1999, Database Language SQL -- Part 4: Persistent Stored Modules (SQL/PSM)*
  - SQL 프로시듀어 제어 명령문에 대한 1999 ISO 표준 정의를 포함하고 있습니다.
- *ISO/IEC 9075-5:1999, Database Language SQL -- Part 4: Host Language Bindings (SQL/Bindings)*

– 호스트 언어 바인딩 및 동적 SQL에 대한 1999 ISO 표준 정의를 포함하고 있습니다.

제2장 개념

이 장에서는 SQL에서 일반적으로 사용되는 개념에 대해 간략하게 설명합니다. 이 장의 목적은 상위 관점에서의 개념을 제공하기 위한 것입니다. 계속되는 참조 데이터에서 보다 상세한 관점을 제공합니다.

관계형 데이터베이스

*관계형 데이터베이스*는 테이블의 세트로 인식되어 관계형 데이터 모델로 조작될 수 있는 데이터베이스입니다. 여기에는 데이터를 저장, 관리 및 액세스하는 데 사용되는 오브젝트 세트가 포함됩니다. 이러한 오브젝트의 예로는 테이블, 뷰, 색인, 함수, 트리거 및 패키지가 있습니다.

파티션된 관계형 데이터베이스는 (노드라고 하는) 여러 파티션에서 관리되는 관계형 데이터베이스입니다. 파티션에 걸친 이러한 데이터의 파티션은 대부분의 SQL 문에서 사용자에게 그대로 드러납니다. 그러나, 일부 DDL문은 파티션 정보를 고려합니다(예: CREATE NODEGROUP).

연합 데이터베이스는 데이터가 개별 관계형 데이터베이스와 같은 여러 데이터 소스에 저장되는 관계형 데이터베이스입니다. 데이터는 단일 대형 데이터베이스에 모두 저장되어 있는 것처럼 표시되므로, 관계형 SQL 조회를 통해 액세스할 수 있습니다. 데이터에 대한 변경사항은 명시적으로 적절한 데이터 소스에 보낼 수 있습니다. 47 페이지의 『DB2 연합 시스템』에서 자세한 설명을 참조하십시오.

SQL

SQL은 관계형 데이터베이스에서 데이터를 정의하고 조작하는 데 사용되는 표준화된 언어입니다. 관계형 데이터 모델에 따라, 데이터베이스는 테이블의 세트로 인식되며, 관계는 테이블에서 값으로 표현됩니다. 그리고 데이터는 하나 이상의 기본 테이블에서 얻을 수 있는 결과 테이블을 지정하여 검색합니다.

SQL문은 데이터베이스 관리 프로그램에 의해 실행됩니다. 데이터베이스 관리 프로그램의 기능 중 하나는 결과 테이블의 스펙을 데이터 검색을 최적화할 수 있는 일련의 내부 동작으로 변형시키는 것입니다. 변형은 준비와 바인딩이라는 두 단계로 일어납니다.

실행할 수 있는 모든 SQL문은 실행되기 전에 준비되어야 합니다. 명령문을 준비하면 명령문의 실행 양식 또는 작동 양식이 됩니다. SQL문의 준비 방식과 작동 양식의 지속성은 정적 SQL인지 동적 SQL인지에 따라 다릅니다.

Embedded SQL

Embedded SQL문은 C 같은 응용프로그램 언어 내에서 쓰여지고, 응용프로그램을 컴파일하기 전에 SQL 사전 처리 컴파일에 의해 사전 처리 컴파일되는 SQL문입니다. Embedded SQL에는 동적과 정적의 두 가지 유형이 있습니다.

정적 SQL

정적 SQL문의 원시 양식은 COBOL 같은 호스트 언어로 쓰여진 응용프로그램에 내장됩니다. 명령문은 프로그램이 실행되기 전에 준비되어, 프로그램 실행 이후에도 명령문의 작동 양식이 유지됩니다.

정적 SQL문이 들어 있는 원시 프로그램은 이것을 컴파일하기 전에 SQL 사전 처리 컴파일러에 의해 처리되어야 합니다. 사전 처리 컴파일러는 SQL문을 호스트 언어 주석으로 바꾼 후, 데이터베이스 관리 프로그램을 호출하는 호스트 언어 명령문을 작성합니다. SQL문의 구문은 사전 처리 컴파일 중에 점검됩니다.

SQL 응용프로그램의 준비에는 사전 처리 컴파일, 정적 SQL 문의 목표 데이터베이스로의 바인딩 및 수정된 원시 프로그램의 컴파일이 포함됩니다. 위의 단계는 *응용프로그램 개발 안내서*에 설명되어 있습니다.

동적 SQL

Embedded 동적 SQL문을 포함하는 프로그램은 정적 SQL문을 포함하는 프로그램처럼 사전 처리 컴파일되어야 하나, 정적 SQL문과는 달리 동적 SQL문은 런타임에 구성되고 준비됩니다. SQL문 텍스트는 PREPARE 및 EXECUTE문 또는

EXECUTE IMMEDIATE문을 사용하여 준비되고 실행됩니다. 명령문이 SELECT 문인 경우, 커서 조작으로 수행될 수도 있습니다.

DB2 콜 레벨 인터페이스(CLI) & ODBC(Open Database Connectivity)

DB2 콜 레벨 인터페이스(DB2 CLI)는 하나의 API이며, 이 안에서 함수가 응용프로그램에 제공되어 동적 SQL문을 처리합니다. CLI 프로그램은 Microsoft나 다른 벤더로부터 사용할 수 있는 ODBC(Open Database Connectivity) Software Developer's Kit(SDK)을 통해 컴파일하여 ODBC 데이터 소스에 액세스할 수 있습니다. Embedded SQL을 사용할 때와는 달리 사전 처리 컴파일이 필요없습니다. 이 인터페이스를 사용하여 개발된 응용프로그램은 각각의 데이터베이스에 대해 컴파일하지 않고도 다양한 데이터베이스를 액세스할 수 있습니다. 이 인터페이스를 통해, 응용프로그램은 실행시 프로시듀어 호출을 사용하여 데이터베이스에 연결하고, SQL문을 실행하고, 리턴된 데이터와 상태 정보를 얻을 수 있습니다.

DB2 CLI 인터페이스는 Embedded SQL에서는 사용할 수 없는 여러 가지 기능을 제공합니다. 몇 가지 예를 들면 다음과 같습니다.

- CLI에서는 DB2 데이터베이스 관리 시스템 계열들간에서 데이터베이스 시스템 카탈로그 정보를 조회하고 검색하는 일관성있는 방법을 지원하는 함수 호출을 제공합니다. 이것으로 데이터베이스 서버에 따라 카탈로그 조회를 작성할 필요가 줄어 들었습니다.
- CLI는 커서를 통해 화면이동할 수 있는 기능을 제공합니다.
  - 하나 이상의 행씩 전방향으로.
  - 하나 이상의 행씩 후방향으로.
  - 하나 이상의 행씩 첫번째 행으로부터 전방향으로.
  - 하나 이상의 행씩 마지막 행으로부터 후방향으로.
  - 커서에서 이전에 저장된 위치로부터.
- CLI를 이용해 작성된 응용프로그램에서 호출된 저장 프로시듀어에서는 이들 프로그램에 결과 집합을 리턴할 수 있습니다.

*CLI Guide and Reference*에서는 이 인터페이스에서 지원하는 API에 대해 설명합니다.

Java Database Connectivity(JDBC) 및 Embedded SQL for Java(SQLJ) 프로그램

DB2 Universal Database는 Java Database Connectivity (JDBC) 및 embedded SQL for Java(SQLJ), 이렇게 두 개의 표준에 기반한 Java 프로그래밍 API를 구현합니다. 둘다 DB2를 액세스하는 Java 응용프로그램 및 애플릿을 작성하는 데 사용할 수 있습니다.

JDBC 호출은 Java 원시 메소드를 통해 DB2 CLI에 대한 호출로 변역됩니다. JDBC 요청은 DB2 클라이언트로부터 DB2 CLI를 통해 DB2 서버로 흐릅니다. 정적 SQL은 JDBC에 의해 사용될 수 없습니다.

SQLJ 응용프로그램은 데이터베이스에 연결하고 SQL 오류를 처리하는 것과 같은 타스크를 위한 기초로서 JDBC를 사용하지만, SQLJ 소스 파일에 embedded 정적 SQL문을 포함할 수도 있습니다. 그 결과 나오는 Java 소스 코드를 컴파일하기 전에 SQLJ 번역기로 SQLJ 소스 파일을 번역해야 합니다.

JDBC 및 SQLJ 응용프로그램에 대한 *응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.

대화식 SQL

대화식 SQL문은 명령행 처리기나 명령 센터와 같은 인터페이스를 통해 사용자가 입력합니다. 이러한 명령문들은 동적 SQL문으로 처리됩니다. 예를 들면, 대화식 SELECT문은 DECLARE CURSOR, PREPARE, DESCRIBE, OPEN, FETCH 및 CLOSE문을 사용하여 동적으로 처리할 수 있습니다.

*Command Reference*에는 명령행 처리기(CLP)나 유사한 장치 및 제품을 사용하여 실행할 수 있는 명령들이 나열되어 있습니다.

스키마

*스키마*는 명명된 오브젝트의 콜렉션입니다. 스키마는 데이터베이스 내에서 오브젝트를 논리적으로 분류합니다. 스키마에 포함될 수 있는 오브젝트로는 테이블, 뷰, 별명, 트리거, 함수 및 패키지가 있습니다.

스키마는 데이터베이스 내의 오브젝트이기도 합니다. 이 오브젝트는 사용자를 소유자로 기록한 상태에서 CREATE SCHEMA문을 사용하여 명시적으로 작성됩니다. 이것은 또한 사용자가 IMPLICIT_SCHEMA 권한을 가지고 있을 경우, 다른 오브젝트가 작성될 때 내재적으로 작성될 수 있습니다.

*스키마 이름*은 두 부분으로 된 오브젝트 이름 가운데 상층 부분으로 사용됩니다. 스키마에 포함된 오브젝트는 오브젝트 작성시 해당 스키마에 지정됩니다. 오브젝트가 지정되는 대상 스키마는, 스키마 이름이 특정하게 규정된 경우 오브젝트 이름에 의해 결정되고, 스키마 이름이 규정되지 않은 경우 기본 스키마 이름에 의해 결정됩니다.

예를 들어, DBADM 권한을 가지고 있는 사용자가 사용자 A를 위해 C라고 하는 스키마를 작성합니다.

```
CREATE SCHEMA C grant A
```

그러면, 사용자 A는 다음 명령문을 실행하여 스키마 C에 X라는 테이블을 작성할 수 있습니다.

```
CREATE TABLE C.X (COL1 INT)
```

스키마 사용의 제어

데이터베이스가 작성될 때, 모든 사용자가 IMPLICIT_SCHEMA 권한을 가지고 있습니다. 그러면, 사용자는 아직 존재하지 않는 임의의 스키마에 오브젝트를 작성할 수 있습니다. 내재적으로 작성된 스키마를 사용하면 사용자가 이 스키마에서 다른 오브젝트를 작성할 수 있습니다.[1]

1. 내재적으로 작성되는 스키마에 대한 기본 특권은 이전 버전과의 상향 호환성을 제공합니다. 별명, 구별 유형, 기능 및 트리거 생성이 내재적으로 작성된 스키마로 확대됩니다.

IMPLICIT_SCHEMA 권한이 PUBLIC으로부터 권한 취소될 경우, 스키마는 CREATE SCHEMA 명령문을 사용하여 명시적으로 작성되거나 IMPLICIT_SCHEMA 권한이 부여된 사용자(DBADM 권한이 있는 사용자와 같은)에 의해 내재적으로 작성됩니다. PUBLIC에서 IMPLICIT_SCHEMA 권한을 취소하면 스키마 이름 사용에 대한 제어가 증가되지만, 오브젝트를 작성하려고 할 때 기존의 응용프로그램에서 권한 부여 오류가 발생할 수 있습니다.

스키마에서 오브젝트를 작성, 변경 및 삭제할 권한 소유자를 제어하는 스키마와 연관된 특권도 있습니다. 스키마 소유자에게는 일차적으로 스키마에 대해 이들 모든 특권이 주어지며 이 특권을 다른 사용자에게 부여하기 위한 권한도 주어집니다. 암시적으로 작성된 스키마는 시스템의 소유이며, 일차적으로는 모든 사용자에게 이러한 스키마에 오브젝트를 작성할 특권이 주어집니다. DBADM 또는 SYSADM 권한을 가지고 있는 사용자는 어떤 스키마에 대해 갖는 특권도 변경할 수 있습니다. 그러므로, 임의의 스키마(명시적으로 작성된 스키마도 포함)에 있는 오브젝트를 작성, 변경 및 삭제하기 위한 액세스를 제어할 수 있습니다.

테이블

테이블은 데이터베이스 관리 프로그램에 의해 유지보수되는 논리적인 구조입니다. 테이블은 컬럼과 행으로 구성됩니다. 행은 테이블 내에서 순서를 지정할 필요가 없습니다(순서는 응용프로그램에서 결정합니다.). 모든 컬럼과 행의 교차점에는 *값*이라는 특정 데이터 항목이 있습니다. *컬럼*은 같은 유형의 값 세트이거나 부속 유형 중 하나입니다. *행*은 일련의 값으로, *n*번째 값은 테이블의 *n*번째 컬럼의 값을 나타냅니다.

*기본 테이블*은 CREATE TABLE문으로 작성되어 사용자 데이터를 지속적으로 유지하는 데 사용됩니다. *결과 테이블*은 데이터베이스 관리 프로그램이 하나 이상의 기본 테이블에서 선택하거나 생성한 조회를 만족시키는 행들의 집합입니다.

요약 테이블은 조회에 의해 정의되며 테이블 내의 데이터를 판별하는 데도 사용되는 테이블입니다. 요약 테이블은 조회의 성능을 향상시키는 데도 사용될 수 있습니다. 데이터베이스 관리 프로그램이 조회의 일부분이 요약 테이블을 사용하여 해결될 수 있다고 판단할 경우, 조회는 데이터베이스 관리 프로그램에 의해 요약 테

이블을 사용하도록 다시 쓰여집니다. 이 결정은 CURRENT REFRESH AGE 및 CURRENT QUERY OPTIMIZATION 특수 레지스터와 같은 특정의 설정을 근거로 합니다.

테이블에는 개별적으로 정의된 각 컬럼의 데이터 유형이나, 사용자 정의 구조화 유형 속성에 기초한 컬럼 유형이 포함될 수 있습니다. 이를 *분류된 테이블(typed table)*이라고 합니다. 사용자 정의 구조화 유형은 유형 계층의 일부입니다. *부속 유형(subtype)*은 *상위 유형(supertype)*으로부터 속성을 물려받습니다. 마찬가지로, 유형화된 테이블은 테이블 계층의 일부가 될 수 있습니다. *서브테이블(subtable)*은 *상위 테이블*으로부터 컬럼을 물려받습니다. *부속 유형*이란, 사용자가 정의한 구조화 유형과, 유형 계층상 그 아래에 있는 모든 사용자 정의 구조화 유형을 가리킵니다. 구조화 유형 T의 *적절한 부속 유형*은 유형 계층상 T 아래에 있는 구조화 유형입니다. 마찬가지로 *서브테이블*은, 분류된 테이블과, 테이블 계층상 그 아래에 있는 모든 분류 테이블을 가리킵니다. T 테이블의 *적절한 서브테이블*은 테이블 계층상 T 아래에 있는 테이블을 말합니다.

*선언된 임시 테이블*은 DECLARE GLOBAL TEMPORARY TABLE 명령문으로 작성되고 단일 응용프로그램 대신 임시 데이터를 보유하기 위해 사용됩니다. 이 테이블은 응용프로그램이 데이터베이스로부터 연결해제할 때 내재적으로 삭제됩니다.

뷰

*뷰*는 하나 이상의 테이블에 있는 데이터를 보는 다른 방법을 제공합니다.

뷰는 결과 테이블의 명명된 스펙입니다. 스펙은 뷰가 SQL문에서 참조될 때마다 실행되는 하나의 SELECT문입니다. 그러므로, 뷰는 기본 테이블처럼 컬럼과 행이 있는 것으로 생각할 수 있습니다. 검색을 위해, 모든 뷰들은 기본 테이블처럼 사용될 수 있습니다. 뷰가 삽입, 갱신 또는 삭제 작업에서 사용될 수 있는지의 여부는 CREATE VIEW의 설명에서 기술된 것과 같이 그 정의에 따라 달라집니다(931 페이지의 『CREATE VIEW』에서 자세한 정보를 참조하십시오.).

뷰의 컬럼이 기본 테이블의 컬럼에서 직접 얻은 것인 경우, 컬럼에는 기본 테이블 컬럼에 적용되는 제한사항이 그대로 적용됩니다. 예를 들면, 뷰에 기본 테이블의 외부 키(foreign key)가 있는 경우, 이 뷰를 사용하는 INSERT와 UPDATE 조작

은 기본 테이블과 같은 참조 제한조건의 적용을 받습니다. 또한 뷰의 기본 테이블이 상위 테이블인 경우, 이 뷰를 사용하는 DELETE와 UPDATE 조작은 기본 테이블에서의 DELETE와 UPDATE 조작과 같은 규칙이 적용됩니다.

뷰에는 결과 테이블로부터 도출된 각 컬럼의 데이터 유형이나, 사용자가 정의한 구조화 유형의 속성에 기초한 컬럼 유형이 포함됩니다. 이를 *분류된 뷰*라고 합니다. 분류된 테이블과 마찬가지로 분류된 뷰는 뷰 계층의 일부가 될 수 있습니다. *부속 뷰*는 *상위 뷰*로부터 컬럼을 물려받습니다. *부속 뷰*는 분류된 테이블과, 뷰 계층상 그 아래에 있는 모든 분류된 뷰를 가리킵니다. V 뷰의 *적절한 부속 뷰*는 분류된 뷰 계층상 V 아래에 있는 뷰입니다.

뷰는 사용 불가능해 질 수 있으며, 이 경우 더 이상 SQL문에 사용할 수 없습니다.

별명

*별명(alias)*은 테이블이나 뷰의 다른 이름입니다. 기존 테이블이나 뷰가 참조될 수 있는 상황에서 테이블이나 뷰를 참조하는 데 사용할 수 있습니다.[2] 테이블 및 뷰와 같이, 별명은 작성, 삭제하고 주석을 연관시킬 수 있습니다. 별명(nicknames)에 대해서는 별명을 작성할 수 있습니다. 테이블과는 달리, 별명은 연쇄(chaining)라고 하는 프로세스에서 서로 참조할 수 있습니다. 별명은 공개적으로 참조되는 이름이므로, 별명을 사용하는 데 특별한 권한이 필요하지 않습니다. 그러나, 별명으로 참조하는 테이블과 뷰를 액세스하려면 현재 문맥에 대한 적합한 권한이 필요합니다.

테이블 별명 이외에, 데이터베이스 및 네트워크 별명 같은 다른 유형의 별명이 있습니다.

82 페이지의 『별명』 및 627 페이지의 『CREATE ALIAS』에서 자세한 정보를 참조하십시오.

2. 별명이 모든 컨텍스트에서 사용될 수 있는 것은 아닙니다. 예를 들어, 점검 제한조건의 점검조건에는 사용할 수 없습니다. 또한, 별명은 선언된 임시 테이블을 참조할 수 없습니다.

색인

*색인*은 기본 테이블의 행을 나타내는 포인터의 정렬된 세트입니다. 각 색인은 하나 이상의 테이블 컬럼에 있는 데이터값을 기준으로 합니다. 색인은 테이블에 있는 데이터를 분리시키는 오브젝트입니다. 색인이 작성되면, 데이터베이스 관리 프로그램은 이 구조를 구성한 후, 자동으로 이것을 유지보수합니다.

색인은 다음을 위해 데이터베이스 관리 프로그램에 의해 사용됩니다.

- 성능 향상. 대부분의 경우, 데이터 액세스는 색인이 없는 것보다 속도가 빨라집니다.

  색인은 뷰에 대해 작성할 수 없습니다. 그러나, 뷰가 기본으로 하는 테이블에 대해 작성된 색인은 뷰에 대한 조작의 성능을 향상시킵니다.
- 고유성 보장. 고유 색인이 있는 테이블에서는 같은 키가 있는 행이 있을 수 없습니다.

키

*키*는 특정 행에 액세스하거나 이를 식별하는 데 사용할 수 있는 컬럼 세트입니다. 키는 테이블, 색인 또는 참조 제한조건의 설명에서 식별됩니다. 한 컬럼이 두 개 이상의 키의 일부가 될 수 있습니다.

둘 이상의 컬럼으로 구성된 키를 *복합 키(composite key)*라고 합니다. 복합 키가 있는 테이블에서, 복합 키 내의 컬럼 순서는 테이블에서의 순서에 제한을 받지 않습니다. 복합 키에 대해 사용될 때 *값*은 복합 값을 나타냅니다. 그러므로, 『외부 키의 값은 기본 키(primary key)의 값과 같아야 한다』라는 규칙은 외부 키 값의 각 구성요소는 기본 키 값의 해당 요소와 일치해야 함을 의미합니다.

고유 키

*고유 키(unique key)*는 서로 같은 값이 없도록 제한받는 키입니다. 고유 키 컬럼에는 널(NULL) 값이 들어갈 수 없습니다. 이러한 제한조건은 INSERT 또는 VPDATE 같은 데이터 값 변경 조작의 시행 중 데이터베이스 관리 프로그램을 통해 적용됩니다. 제한조건을 적용하는 데 사용되는 기법을 *고유 색인*이라고 합니다.

그러므로, 모든 고유 키는 고유 색인의 키입니다. 이러한 색인에 대해 UNIQUE 속성을 가졌다고도 합니다. 19 페이지의 『고유성 제한조건』에서 자세한 내용을 참조하십시오.

기본 키

*기본 키*는 고유 키의 특수한 경우입니다. 테이블에는 하나 이상의 기본 키가 있을 수 없습니다. 17 페이지의 『고유 키』에서 좀더 자세한 설명을 참조하십시오.

외부 키

*외부 키(foreign key)*는 참조 제한조건의 정의에 지정되는 키입니다. 20 페이지의 『참조 제한조건』에서 자세한 내용을 참조하십시오.

파티션 키

*파티션 키(partitioning key)*는 파티션된 데이터베이스에 있는 테이블 정의의 일부인 키입니다. 파티션 키는 데이터 행이 저장된 파티션을 결정하는 데 사용됩니다. 파티션 키가 정의된 경우, 고유 키와 기본 키가 파티션 키로 동일한 컬럼에 들어 있어야 합니다(컬럼이 여러 개일 수 있습니다.). 테이블에는 하나 이상의 파티션 키가 있을 수 없습니다.

제한조건

*제한조건(constraints)*은 데이터베이스 관리 프로그램이 실행하는 규칙입니다.

제한조건에는 다음의 세 가지 유형이 있습니다.

- *고유성 제한조건(unique constraint)*은 한 테이블내의 둘 이상 컬럼에서 동일한 값을 사용할 수 없도록 하는 규칙입니다. 고유한 기본 키는 지원되는 고유성 제한조건입니다. 예를 들어, 동일한 공급자 식별자를 두 공급자에게 제공하지 않기 위해 공급자 테이블의 공급자 식별자에 대해 고유성 제한조건을 정의할 수 있습니다.
- *참조 제한조건(referential constraint)*은 하나 이상의 테이블 내에 있는 하나 이상의 컬럼 값에 관한 논리적인 규칙입니다. 예를 들면, 테이블 세트는 회사의 공급자에 관한 정보와, 때때로 공급자의 이름 변경사항을 공유합니다. 참조 제한조건은 테이블에 있는 공급자 ID가 공급자 정보에 있는 공급자 ID와 일치해야

만 정의될 수 있습니다. 이러한 제한조건은 결과적으로 공급자 정보를 손실하게 되는 삽입, 갱신, 삭제를 막을 수 있습니다.

- *테이블 점검 제한조건(table check constraint)*에서는 특정 테이블에 데이터를 추가할 때 제한조건을 설정합니다. 예를 들면, 개인 정보를 포함하고 있는 테이블에 봉급 데이터가 추가되거나 갱신될 때 사원에 대한 봉급 레벨은 $20,000.00 이하가 절대 될 수 없다는 것을 정의할 수 있습니다.

참조 제한조건 및 테이블 점검 제한조건은 온(on) 또는 오프(off)시킬 수 있습니다. 대량의 데이터를 데이터베이스에 적재하는 경우가 제한조건의 실행 점검을 off 시킬 수 있는 전형적인 시기입니다. 제한조건을 온 또는 오프로 설정하는 것에 관한 자세한 설명은 1160 페이지의 『SET INTEGRITY』에서 논의됩니다.

고유성 제한조건

*고유성 제한조건*은 키 값이 테이블 내에서 고유할 경우에만 유효하도록 하는 규칙입니다. 고유성 제한조건은 생략 가능하며, PRIMARY KEY절이나 UNIQUE절을 사용하여 CREATE TABLE 또는 ALTER TABLE문에서 정의할 수 있습니다. 고유성 제한조건에 지정되는 컬럼은 NOT NULL로 정의해야 합니다. 고유성 제한조건의 컬럼 변경시 키의 고유성을 보장하기 위해 데이터베이스 관리 프로그램이 고유 색인을 사용합니다.

테이블에는 수에 관계없이 고유성 제한조건이 있을 수 있으며, 이들 중 하나만 기본 키로 정의해도 됩니다. 테이블에는 같은 컬럼 세트에서 하나 이상의 고유 제한조건이 있을 수 없습니다.

참조 제한조건의 외부 키가 참조하는 고유성 제한조건을 *상위 키*라고 합니다.

고유성 제한조건이 CREATE TABLE 문에 정의되면, 데이터베이스 관리 프로그램에 의해 고유 색인이 자동으로 작성되고, 이는 기본 또는 고유한 시스템 필수 색인으로 지정됩니다.

고유성 제한조건이 ALTER TABLE 문에 정의되고 동일한 컬럼에 색인이 존재할 경우, 이 색인이 고유 색인 및 시스템 필수 색인으로 지정됩니다. 이러한 색인이 존재하지 않을 경우, 데이터베이스 관리 프로그램에 의해 고유 색인이 자동으로 작성되어 기본 또는 고유한 시스템 필수 색인으로 지정됩니다.

고유성 제한조건의 정의와 고유 색인의 작성에는 차이가 있다는 점을 유의하십시오. 둘다 고유성을 강제할 수 있지만, 고유 색인은 널(NULL) 입력 가능한 컬럼을 허용하는 반면 일반적으로 상위 키로 사용할 수는 없습니다.

참조 제한조건

*참조 무결성(referential integrity)*은 모든 외부 키의 값이 유효할 때의 데이터베이스 상태입니다. *외부 키*는 그 값이 상위 테이블 행의 적어도 한 기본 키 또는 고유 키 값과 일치해야 하는 테이블의 컬럼이나 컬럼 세트입니다. *참조 제한조건*은 외부 키 값이 다음과 같은 경우에만 유효한 규칙입니다.

- 상위 키의 값으로 표시되는 경우 또는
- 외부 키의 구성요소가 널(NULL)인 경우.

상위 키가 들어 있는 테이블을 참조 제한조건의 *상위 테이블*이라고 하며, 외부 키가 들어 있는 테이블을 테이블에 *종속*된다고 합니다.

참조 제한조건은 생략 가능하며, CREATE TABLE문과 ALTER TABLE문에서 정의할 수 있습니다. 참조 제한조건은 INSERT, UPDATE, DELETE, ALTER TABLE ADD CONSTRAINT 및 SET INTEGRITY문을 수행하는 동안 데이터베이스 관리 프로그램을 통해 적용됩니다. 이것은 명령문 컴파일시 효과적으로 적용됩니다.

RESTRICT의 삭제 또는 갱신 규칙이 있는 참조 제한조건은 모든 다른 참조 제한조건 보다 먼저 적용됩니다. NO ACTION의 삭제 또는 갱신 규칙이 있는 참조 제한조건은 대부분의 경우 RESTRICT처럼 작동합니다. 그러나, 특정 SQL문에서는 차이가 있을 수 있습니다.

참조 무결성, 점검 제한조건 및 트리거는 수행시 서로 결합될 수 있습니다. 이들 요소의 결합에 대한 1451 페이지의 『부록J. 트리거와 제한조건의 상호 작용』에서 자세한 내용을 참조하십시오.

참조 무결성 규칙에는 다음의 개념과 용어가 수반됩니다.

상위 키 참조 제한조건의 기본 키 또는 고유 키.

상위 행 종속 행이 적어도 하나 이상 있는 행.

상위 테이블 참조 제한조건의 상위 키를 포함하는 테이블. 테

이블은 임의 수의 참조 제한조건에서 상위일 수 있습니다. 참조 제한조건에서 상위인 테이블이 참조 제한조건의 종속일 수도 있습니다.

종속 테이블 정의에서 최소한 하나의 참조 제한조건을 포함하는 테이블. 테이블은 임의 수의 참조 제한조건에서 종속일 수 있습니다. 참조 제한조건에서 종속인 테이블이 참조 제한조건의 상위일 수도 있습니다.

하위 테이블 테이블이 테이블T의 종속이거나 T의 종속의 하위인 경우, 테이블 T의 하위라고 합니다.

종속 행 적어도 하나의 상위 행이 있는 행.

하위 행 행이 행 p의 종속이거나 p의 종속의 자인 경우, 행 p의 하위라 합니다.

참조 순환 집합 내의 각 테이블이 자신의 하위인 것과 같은 참조 제한조건의 집합.

자체 참조 행 자신의 상위인 행.

자체 참조 테이블 같은 참조 제한조건에서 상위이고 종속(dependent)인 테이블. 이때 그 제한조건을 *자체 참조 제한조건(self-referencing constraint)*이라고 합니다.

삽입 규칙

참조 제한조건의 삽입 규칙은 외부 키의 널(NULL)이 아닌 삽입 값이 상위 테이블에 있는 상위 키값과 일치해야 한다는 것입니다. 복합 외부 키값은 값 구성요소가 널(NULL)인 경우 널(NULL)입니다. 외부 키가 지정되면 이 규칙은 암시적입니다.

갱신 규칙

참조 제한조건의 갱신 규칙은 참조 제한조건이 정의될 때 지정됩니다. 선택사항은 NO ACTION과 RESTRICT입니다. 갱신 규칙은 상위 테이블의 행 또는 종속 테이블의 행이 갱신되는 경우에 적용됩니다.

상위 행에서, 상위 키 컬럼의 값이 갱신되는 경우

- 종속 테이블의 임의의 행이 키의 원래 값과 일치하면, 갱신 규칙이 RESTRICT 일 때 갱신이 거부됩니다.
- 갱신 명령문이 완료되었을 때(트리거 이후는 제외) 종속 테이블의 임의 행에 해당하는 상위 키가 없는 경우, 갱신 규칙이 NO ACTION이면 갱신이 거부됩니다.

종속 행에서, 외부 키가 지정될 때, 내재된 갱신 규칙은 NO ACTION이 됩니다. NO ACTION은 널(null)이 아닌 값으로 갱신되는 외부 키가 반드시 상위 테이블의 상위 키의 일부 값과 일치해야 함을 의미합니다.

복합 외부 키값은 값 구성요소가 널(NULL)인 경우 널(NULL)입니다.

삭제 규칙

참조 제한조건의 삭제 규칙은 참조 제한조건이 정의될 때 지정됩니다. 선택사항은 NO ACTION, RESTRICT, CASCADE 또는 SET NULL입니다. SET NULL은 일부 외부 키 컬럼에서 널(NULL)값이 허용될 경우에만 지정할 수 있습니다.

참조 제한조건의 삭제 규칙은 상위 테이블의 행이 삭제될 때 적용됩니다. 좀더 정확하게, 규칙은 상위 테이블의 행이 삭제되거나 전달된 삭제 조작(아래에서 정의됨)의 오브젝트이고, 행이 참조 제한조건의 종속 테이블에서 종속인 경우 적용됩니다. P를 상위 테이블, D를 종속 테이블, p를 삭제되거나 전달된 조작의 오브젝트라고 합시다. 각 삭제 규칙에 대해 다음이 적용됩니다.

- RESTRICT 또는 NO ACTION: 오류가 발생하고 행이 삭제되지 않습니다.
- CASCADE: 삭제 조작이 D에 있는 p의 종속 테이블로 전달됩니다.
- SET NULL: D에 있는 각 P의 종속 테이블의 외부 키에 있는 널(NULL) 입력 가능 컬럼이 널(NULL)로 설정됩니다.

테이블이 상위인 각각의 참조 제한조건에는 자신의 삭제 규칙이 있고, 모든 적용할 수 있는 삭제 규칙은 삭제 조작의 결과를 결정하는 데 사용됩니다. 그러므로, 삭제 규칙이 RESTRICT나 NO ACTION인 참조 제한조건에서 종속이거나, 삭제 규칙이 RESTRICT나 NO ACTION인 참조 제한조건에서 종속으로 삭제 조작이 연쇄되는 경우, 행은 삭제할 수 없습니다.

상위 테이블 P에서의 행 삭제에는 다른 테이블이 포함되며, 이러한 테이블의 행에도 영향을 미칩니다.

- 테이블 D가 P의 종속이고 삭제 규칙이 RESTRICT나 NO ACTION인 경우, D가 이 조작에 포함되기는 하지만 조작에 영향을 받지는 않습니다.
- D가 P의 종속이고 삭제 규칙이 SET NULL인 경우, D는 이 조작에 포함되고 D 행은 조작동안 갱신됩니다.
- D가 P의 종속이고 삭제 규칙이 CASCADE인 경우, D는 이 조작에 포함되고 D 행은 조작동안 삭제됩니다.

  D의 행이 삭제되면, P에 대한 삭제 조작이 D에 전달됩니다. D 역시 상위 테이블이면, 이 목록에서 기술된 조치가 D의 종속에 전달됩니다.

P에서의 삭제 조작에 포함된 테이블은 P에 *연속 삭제(delete-connected)*되었다고 합니다. 그러므로, 테이블은 P나, P 연쇄로부터 조작을 삭제하는 테이블에 종속될 경우 테이블 P에 대해 연속 삭제됩니다.

테이블 점검 제한조건

*테이블 점검 제한조건(table check constraint)*은 테이블 각 행의 하나 이상의 컬럼에 허용된 값을 지정하는 규칙입니다. 이 규칙은 선택적이며 CREATE TABLE 및 ALTER TABLE과 같은 SQL문을 이용하여 정의할 수 있습니다. 테이블 점검 제한조건의 스펙은 검색 조건의 제한된 양식입니다. 제약 중 하나는 테이블 *T* 상의 테이블 점검 제한조건에 있는 컬럼 이름은 *T*의 컬럼을 식별해야 하는 것입니다.

테이블에는 임의의 수의 테이블 점검 제한조건이 있을 수 있습니다. 다음 경우에 시행됩니다.

- 테이블에 행이 삽입될 때
- 테이블의 행이 갱신될 때

테이블 점검 제한조건은 삽입 또는 갱신되는 각 행에 검색 조건을 적용하여 시행됩니다. 오류는 검색 조건의 결과가 모든 행에 대해 실패하는 경우에 발생합니다.

하나 이상의 테이블 점검 제한조건이 기존 데이터가 있는 테이블의 ALTER TABLE문에 정의되어 있으면, ALTER TABLE문이 수행되기 전에 새로운 조건

에 대해 기존 데이터를 점검합니다. 테이블은 *점검 보류(check pending) 상태*가 될 수 있는 데, 이 상태에서는 ALTER TABLE문이 데이터를 점검하지 않고 성공할 수 있습니다. SET INTEGRITY문은 테이블을 점검 보류 상태로 만드는 데 사용됩니다. 또한 제한사항에 대해 각 행의 점검을 재시도할 때에도 사용됩니다.

트리거

*트리거*는 특정 테이블에 대한 삭제, 삽입 또는 갱신 조작시 실행되는 또는 이러한 조작으로 트리거되는 조치 집합을 정의합니다. 그러한 SQL 조작이 수행되면, 트리거가 *활성화*되었다고 합니다.

트리거는 참조 제한조건 및 점검 제한조건과 함께 사용되어 데이터 무결성 규칙이 적용됩니다. 트리거는 또한 다른 테이블을 갱신하거나, 삽입 또는 갱신된 행에 대한 값을 자동으로 생성하거나 변형하는 데 또는 경보 같은 타스크를 수행하는 함수를 호출하는 데 사용할 수 있습니다.

트리거는 여러 가지 상태의 데이터를 포함하는 규칙인 *과도기적* 비지니스 규칙(예, 급여를 10% 이상 올릴 수 없음)을 정의하고 실행하는 유용한 기법입니다. 데이터 상태가 한 가지인 데이터가 들어 있는 규칙의 경우, 점검 제한조건과 참조 무결성 제한조건이 고려되어야 합니다.

트리거를 사용하면 데이터베이스에 있는 비지니스 규칙을 수행하도록 하는 논리를 제공하여, 테이블을 사용하는 응용프로그램이 이것을 강제로 수행하지 않아도 됩니다. 모든 테이블상에서 시행되는 중심 논리는, 논리가 변경될 때 응용프로그램을 변경하지 않아도 되므로 유지보수를 보다 쉽게 할 수 있다는 의미입니다.

트리거는 선택사항으로 CREATE TRIGGER문을 사용하여 정의됩니다.

트리거가 활성화되어야 할 시기를 결정하는 데 사용되는 다수의 기준이 있습니다.

- *주제 테이블*은 트리거가 정의될 테이블을 정의합니다.
- *트리거 이벤트(event)*는 주제 테이블을 수정하는 특정 SQL 조작을 정의합니다. 이 조작은 삭제, 삽입 또는 갱신할 수 있습니다.
- *트리거 활성화 시간*은 트리거 이벤트가 주제 테이블에서 수행되기 전에 또는 후에 트리거가 활성화되어야 하는지를 결정합니다.

트리거가 활성화되도록 하는 명령문에는 *영향받은 행 집합(set affected rows)*이 들어갑니다. 이들 삭제, 삽입 또는 갱신되는 주제 테이블의 행입니다. *트리거 수준(granularity)*도 트리거의 조치가 명령문에 대해 한번 수행될 것인지 또는 영향받은 행 집합에 있는 각 행에 대해 한번씩 수행될 것인지를 결정합니다.

*트리거 조치*는 선택적인 검색 조건과 트리거가 활성화될 때마다 실행되는 일련의 SQL문으로 구성됩니다. SQL문은 검색 조건이 참으로 평가되는 경우에만 수행됩니다. 트리거 활성화 시간이 트리거 이벤트 이전일 경우, 트리거 조치에는 선택하고, 전이 변수를 설정하며, SQLSTATE 신호를 보내는 명령문이 포함될 수 있습니다. 트리거 활성화 시간이 트리거 이벤트 이후일 경우, 트리거 조치에는 선택, 갱신, 삽입, 삭제하고 SQLSTATE 신호를 보내는 명령문이 포함될 수 있습니다.

트리거 조치는 영향받은 행 집합에 있는 값을 참조합니다. 이것은 *전이 변수*의 사용을 통해 지원됩니다. 전이 변수는 기존값(갱신 이전)을 참조할 것인지 또는 새로운 값(갱신 이후)을 참조할 것인지를 결정하는 지정된 이름으로 규정화된 주제 테이블에 있는 컬럼의 이름을 사용합니다. 새로운 값은 또한 갱신이나 삽입 트리거를 수행하기 전에 SET 전이 변수 명령문을 사용하여 변경될 수 있습니다. 영향받은 행 세트에 있는 값을 참조한다는 것은 *전이 테이블*의 사용을 의미합니다. 전이 테이블은 또한 주제 테이블의 컬럼 이름을 사용하지만 영향받은 행 전체 집합이 테이블로 취급되도록 하는 지정된 이름을 갖지는 않습니다. 전이 테이블은 트리거 이후에만 사용될 수 있으며, 별도의 전이 테이블을 이전 값과 새 값에 대해 정의할 수 있습니다.

다수의 트리거가 테이블, 이벤트 또는 활성화 시간의 조합으로 지정될 수 있습니다. 트리거가 활성화되는 순서는 트리거가 작성된 순서와 같습니다. 그러므로, 가장 최근에 작성된 트리거가 가장 마지막으로 활성화됩니다.

트리거의 활성화로 *트리거 연쇄(trigger cascading)*가 발생할 수 있습니다. 이것은 다른 트리거를 활성화시키거나 같은 트리거를 다시 활성화시키는 SQL문을 수행하는 한 트리거의 활성화 결과입니다. 트리거 조치로 인해, 원래 수정의 결과로 또는 추가 트리거를 활성화시키는 참조 무결성 삭제 규칙의 결과로 갱신이 야기될 수도 있습니다. 트리거 연쇄를 사용하여 트리거와 참조 무결성 삭제 규칙의 중요한 체인이 하나의 삭제, 삽입 또는 갱신 명령문의 결과로 데이터베이스에 상당한 변경을 가하도록 활성화될 수 있습니다.

이벤트 모니터

*이벤트 모니터*에서는 이벤트의 결과로 특정 데이터를 추적합니다. 예를 들면, 데이터베이스를 시작하는 것은 이벤트 모니터가 데이터베이스를 사용하는 권한 부여 ID의 한 시간마다의 스냅샷을 취하여 시스템상의 사용자 수를 추적하게 하는 이벤트입니다.

이벤트 모니터는 명령문(SET EVENT MONITOR STATE)으로 활성화 또는 비활성화됩니다. 함수(EVENT_MON_STATE)를 이용하여 현재의 이벤트 모니터의 상태, 즉, 이벤트 모니터가 활성화되어 있는지 또는 비활성되어 있는지를 알 수 있습니다.

조회

*조회(query)*는 (임시)결과 테이블을 지정하는 특정 SQL문들의 구성요소입니다.

테이블 표현식

*테이블 표현식*은 단순한 조회(임시) 결과 테이블을 작성합니다. 절들은 결과 테이블을 더욱 세분화합니다. 예를 들어, 테이블 표현식은 여러 부서의 관리자를 모두 선택하되 15년 이상의 경력이 있으며 뉴욕 지사에 근무하는 사람을 지정하는 조회자가 될 수 있습니다.

공통 테이블 표현식

*공통 테이블 표현식(common table expression)*은 복합 조회내의 임시 뷰와 유사하며, 조회내의 다른 지점에서 참조할 수 있습니다. 예를 들어, 뷰가 작성되지 않도록 뷰 대신 참조할 수 있습니다. 사용할 복합 조회 내의 특정 공통 테이블 표현식은 각각 같은 임시 뷰를 공유합니다.

조회 내에서 공통 테이블 표현식의 재귀적인 사용은 부품표(BOM), 항공 예약 시스템 및 네트워크 계획같은 응용프로그램을 지원하는 데 사용될 수 있습니다. BOM 응용프로그램에서의 예들은 1493 페이지의 『부록M. 순환 예: 부품표(BOM: Bill of Materials)』에 포함되어 있습니다.

패키지

*패키지*는 단일 소스 파일로부터의 모든 *섹션*을 포함하는 오브젝트입니다. 섹션은 SQL문의 컴파일된 형태입니다. 모든 섹션은 한 명령문에 해당되는 반면, 모든 명령문이 꼭 섹션을 가질 필요는 없습니다. 정적 SQL을 위해 작성된 섹션은 SQL문에 대한 바인드 또는 작동 양식으로 간주될 수 있습니다. 동적 SQL을 위해 작성된 섹션은 수행시 사용되는 플레이스홀더 제어 구조로 간주될 수 있습니다. 패키지들은 프로그램 준비중에 생성됩니다. 패키지에 대한 *응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.

카탈로그 뷰

데이터베이스 관리 프로그램은 그 제어하에 있는 데이터에 대한 정보가 들어 있는 일련의 뷰 및 기본 테이블을 유지보수합니다. 이러한 뷰와 기본 테이블들을 모아 *카탈로그*라고 합니다. 카탈로그에는 테이블, 뷰, 색인, 패키지 및 함수와 같은 데이터베이스 내의 오브젝트에 관한 정보가 들어 있습니다.

카탈로그 뷰는 기타 데이터베이스 뷰와 비슷합니다. SQL문은 시스템에 있는 다른 뷰에서 데이터를 검색하는 방식과 같은 방식으로 카탈로그 뷰에 있는 데이터를 보는 데 사용될 수 있습니다. 데이터베이스 관리 프로그램에서는 카탈로그에 항상 데이터베이스 내의 오브젝트에 대한 정확한 설명이 들어 있음을 보장합니다. 갱신 가능한 카탈로그 뷰 세트는 카탈로그에 있는 특정 값을 수정하는 데 사용할 수 있습니다(1282 페이지의 『갱신 가능한 카탈로그 뷰』에서 참조하십시오.).

통계 정보도 카탈로그에 포함됩니다. 통계 정보는 관리자가 실행하는 유틸리티나 적합한 권한이 있는 사용자의 갱신 명령을 통해 갱신됩니다.

카탈로그 뷰는 1281 페이지의 『부록D. 카탈로그 뷰』에서 나열됩니다.

응용프로그램 프로세스, 동시성 및 복구

모든 SQL 프로그램은 *응용프로그램 프로세스* 또는 에이전트의 일부로 실행됩니다. 응용프로그램 프로세스에는 하나 이상의 프로그램 수행이 포함되며, 응용프로그램 프로세스는 데이터베이스 관리 프로그램이 자원과 잠금을 할당하는 단위입니다. 다른 응용프로그램 프로세스에는 다른 프로그램의 수행 또는 같은 프로그램의 다른 수행이 포함될 수 있습니다.

둘 이상의 응용프로그램 프로세스가 동시에 같은 데이터의 액세스를 요청할 수 있습니다. *잠금*은 두 응용프로그램 프로세스가 동시에 같은 데이터행을 갱신할 수 없도록 하는 조건하에서 데이터 무결성의 유지보수에 사용되는 기법입니다.

데이터베이스 관리 프로그램은 한 응용프로그램 프로세스가 수행한 아직 확약되지 않은 변경사항을 다른 사용자가 파악하지 못하도록 잠금을 확보합니다. 데이터베이스 관리 프로그램은 응용프로그램 프로세스가 종료되면 그 프로세스 대신에 확보하여 보유한 모든 잠금을 해제하지만, 응용프로그램 프로세스 자체에서 잠금을 곧 해제하도록 명확하게 요청할 수 있습니다. 이는 작업 단위(UOW) 동안 획득된 잠금을 해제하고 또한 작업 단위(UOW) 동안 취해진 데이터베이스를 확약하는 *확약* 조작을 사용하여 행해집니다.

데이터베이스 관리 프로그램은 응용프로그램 프로세스에 의해 이루어진 확약되지 않은 변경사항을 *되돌리*는 방법을 제공합니다. 이것은 응용프로그램 프로세스의 부분에서 실패할 경우에, 또는 *교착 상태*나 잠금 시간종료 상황에서 필요할 수 있습니다. 그러나, 응용프로그램 프로세스 자체에도 데이터베이스 변경사항을 되돌리도록 요청할 수 있습니다. 이 동작을 *구간 복원*이라고 합니다.

*작업 단위(UOW: unit of work)*는 한 응용프로그램 프로세스 내에서 복구 가능한 일련의 조작입니다. 작업 단위(UOW)는 응용프로그램 프로세스가 시작될 때 초기화됩니다[3]. 작업 단위(UOW)는 또한 이전의 작업 단위(UOW)가 응용프로그램 프로세스의 종료가 아닌 다른 이유로 종료했을 때 시작되기도 합니다. 작업 단위

3. DB2 CLI 및 Embedded SQL은 다중 연결(각각 독립적인 트랜잭션임)을 지원하는 *동시 트랜잭션*이라고 하는 연결 모드를 지원합니다. 응용프로그램은 동일한 데이터베이스에 대해 여러 개의 동시적인 연결을 가질 수 있습니다. 다중 스레드 데이터베이스에 대한 세부사항은 *응용프로그램 개발 안내서*에서 자세한 내용을 참조하십시오.

(UOW)는 확약(commit), 구간 복원 조작 또는 응용프로그램 프로세스의 종료로 인해 끝납니다. 확약이나 구간 복원은 종료한 작업 단위(UOW) 내에서의 데이터베이스 변경사항에만 영향을 미칩니다.

이러한 변경사항이 미확약 상태로 남아 있는 반면, 다른 응용프로그램 프로세스는 그 변경사항을 인식할 수 없어서 변경사항이 백아웃될 수 있습니다.[4] 일단 확약되면, 이러한 데이터베이스 변경사항에 다른 응용프로그램 프로세스가 액세스할 수 있으며 더 이상 구간 복원에 의해 되돌릴 수 없습니다.

응용프로그램 프로세스 대신 데이터베이스 관리 프로그램에 의해 확보된 잠금은 작업 단위(UOW)가 종료할 때까지 보류됩니다. 이 규칙에 대한 예외는, 행 사이에 커서가 이동될 때 잠금이 해제되는 커서 안정성(CS) 분리 레벨과, 잠금이 확보되지 않는 미확약 읽기 레벨입니다(30 페이지의 『분리 레벨』 참조).

작업 단위(UOW)의 시작과 종료는 응용프로그램 프로세스 내에서의 일관성 지점을 정의합니다. 예를 들면, 은행 거래에는 한 구좌에서 다른 구좌로의 자금 이체가 포함될 수 있습니다.

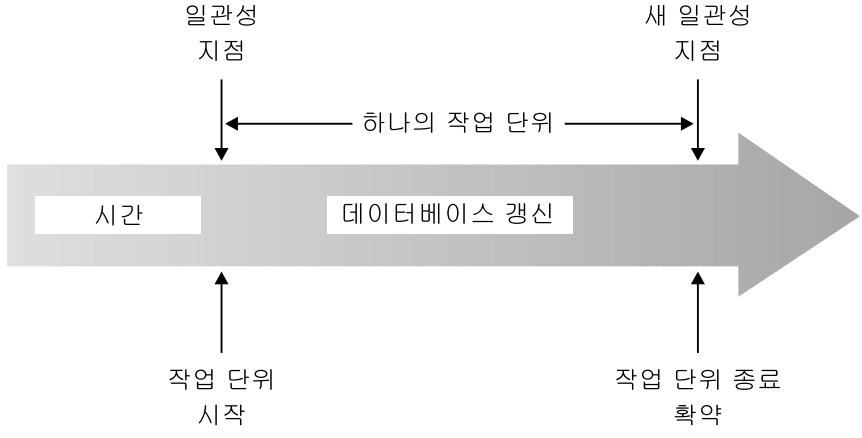


그림 1. 확약 명령문이 있는 작업 단위(UOW)

4. 분리 레벨 미확약 읽기을 제외하고, 33 페이지의 『미확약 읽기(UR, Uncommitted Read)』에 설명되어 있습니다.

그러한 거래에는 이러한 자금이 첫번째 구좌에서 삭제되어 두 번째 구좌로 추가되어야 합니다.

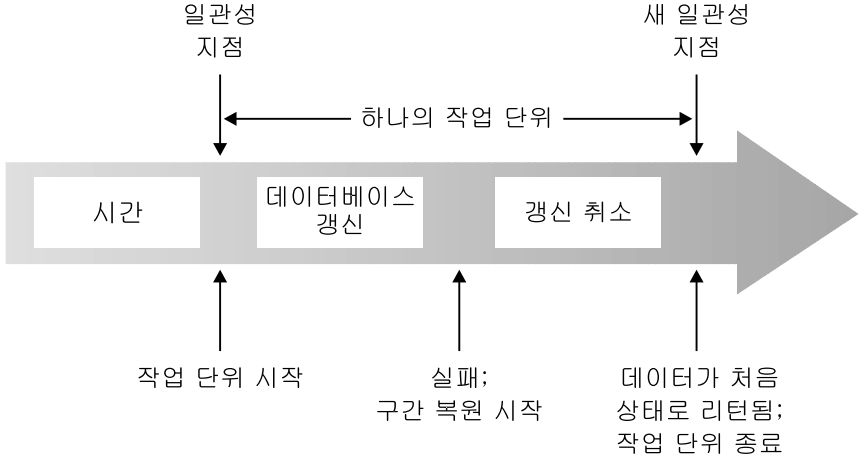


그림 2. 구간 복원 문이 있는 작업 단위(UOW)

빨셈 과정 다음에, 데이터는 일관성이 없게 됩니다. 자금이 두 번째 구좌에 추가되고 난 후에야 일관성이 다시 생깁니다. 두 과정이 완료되면, 확약 조작이 작업 단위(UOW)를 종료하는 데 사용되고, 따라서, 다른 응용프로그램 프로세스에서 변경할 수 있습니다.

작업 단위(UOW)가 종료되기 전에 실패하는 경우, 데이터베이스 관리 프로그램은 확약되지 않은 변경사항을 구간복원하여 작업 단위 시작시 존재했다고 생각되는 데이터의 일관성을 복원합니다.

주: 응용프로그램 프로세스는 자체의 고유한 잠금으로 인해 조작을 수행하는 것을 금지할 수 없습니다.[5]

분리 레벨

응용프로그램 프로세스와 연관되는 *분리 레벨*은 동시적으로 실행되는 다른 응용프로그램 프로세스로부터 응용프로그램 프로세스의 분리 정도를 정의합니다. 응용프로그램 프로세스의 분리 레벨 P는 다음 설명을 지정합니다.

5. 응용프로그램이 동시적인 트랜잭션을 사용할 경우, 한 트랜잭션의 잠금이 동시 트랜잭션의 조작에 영향을 줄 수도 있습니다. *응용프로그램 개발 안내서*에서 자세한 내용을 참조하십시오.

- P에 의해 읽혀지고 갱신된 행을 동시에 수행 중인 다른 응용프로그램 프로세스에서 사용할 수 있는지의 정도.
- 기타 동시 수행 응용프로그램 프로세스의 갱신 활동이 P에 영향을 줄 수 있는지의 정도.

분리 레벨은 패키지의 속성으로 지정되고 그 패키지를 사용하는 응용프로그램 프로세스에 적용됩니다. 분리 레벨은 프로그램 준비 프로세스에서 지정됩니다. 잠금 유형에 따라, 이것은 동시 응용프로그램 프로세스 데이터에 대한 액세스를 제한하거나 금지합니다. 특정 잠금에 대한 다른 유형과 속성에 대해서는 *관리 안내서*에서 자세한 내용을 참조하십시오. 선언된 임시 테이블과 선언된 임시 테이블의 행은 임시 테이블을 선언한 응용프로그램에 의해서만 액세스가 가능하므로 전혀 잠기지 않습니다. 그러므로, 잠금과 분리 레벨에 대한 다음의 사항들은 선언된 임시 테이블에 적용되지 않습니다.

데이터베이스 관리 프로그램은 세 가지의 일반적인 잠금 범주를 지원합니다.

공유 동시 응용프로그램 프로세스를 데이터에 대한 읽기 전용 조작으로 제한합니다.

갱신 동시 응용프로그램 프로세스가 행을 갱신할 수 있는 선언하지 않은 데이터에 읽기 조작만 수행할 수 있도록 제한합니다. 데이터베이스 관리 프로그램은 현재 행에서 볼 수 있는 프로세스가 행을 갱신하는 것으로 가정합니다.

독점 동시 응용프로그램 프로세스가 *미확약 읽기*(읽을 수는 있지만 데이터를 수정할 수는 없음) 분리 레벨을 갖는 응용프로그램 프로세스를 제외한 어떤 방법으로도 데이터에 액세스할 수 없습니다.(33 페이지의 『미확약 읽기(UR, Uncommitted Read)』 참조.)

잠금은 기본 테이블 행에서 발생합니다. 그러나, 데이터베이스 관리 프로그램은 여러 개의 행 잠금을 단일 테이블 잠금으로 대체할 수 있습니다. 이를 *잠금 레벨 자동 업그레이드*라 합니다. 응용프로그램 프로세스에서는 적어도 최소 요청된 잠금 레벨을 보장받습니다.

DB2 Universal Database 데이터베이스 관리 프로그램은 네 가지 분리 레벨을 지원합니다. 분리 레벨에 관계없이, 데이터베이스 관리 프로그램에서는 삽입, 갱신 또는 삭제되는 모든 행에 독점 잠금을 합니다. 따라서, 모든 분리 레벨에서는 작업 단위(UOW)동안 이 응용프로그램 프로세스에서 변경된 행이 작업 단위가 완료할 때까지 어떤 다른 응용프로그램 프로세스에 의해서도 변경되지 않게 합니다. 분리 레벨은 다음과 같습니다.

반복 읽기(RR, Repeatable Read)

RR 레벨은 다음을 보장합니다.

- 작업 단위동안 읽혀진 행은[6] 작업 단위가 완료될 때까지 다른 응용프로그램 의해 변경되지 않습니다.[7]
- 다른 응용프로그램 프로세스가 변경한 행은 그 응용프로그램 프로세스가 확약할 때까지 읽힐 수 없습니다.

RR에서는 가상 행(읽기 안정성 참조) 표시가 허용되지 않습니다.

독점 잠금 외에도, 레벨 RR에서 수행중인 응용프로그램 프로세스는 참조하는 모든 행에 최소한 공유 잠금을 확보합니다. 게다가, 잠금은 응용프로그램 프로세스가 동시 응용프로그램 프로세스의 영향에서 완전하게 분리되도록 수행됩니다.

읽기 안정성(RS, Read Stability)

레벨 RR처럼, 레벨 RS는 다음을 보장합니다.

- 작업 단위(UOW) 동안 읽혀진 행은[6] 작업이 완료될 때까지 다른 응용프로그램 프로세스에 의해 변경되지 않습니다[8].
- 다른 응용프로그램 프로세스가 변경한 행은 그 응용프로그램 프로세스가 확약할 때까지 읽힐 수 없습니다.

---

6. 행은 상응하는 OPEN문과 동일한 작업 단위(UOW) 내에서 읽혀야 합니다. 952 페이지의 『DECLARE CURSOR』의 WITH HOLD를 참조하십시오.

7. CLOSE문에서 선택적인 WITH RELEASE 절을 사용하는 것은, 커서가 다시 열릴 경우 반복할 수 없는 읽기 및 팬텀에 대한 어떤 보증도 더이상 이전에 액세스된 행에 적용되지 않음을 의미합니다.

8. CLOSE문에서 선택적인 WITH RELEASE 절을 사용하는 것은, 커서가 다시 열릴 경우 반복할 수 없는 읽기 및 팬텀에 대한 어떤 보증도 더이상 이전에 액세스된 행에 적용되지 않음을 의미합니다.

RR과는 달리, RS는 동시 응용프로그램 프로세스의 영향에서 응용프로그램 프로세스를 완전히 분리하지 못합니다. 레벨 RS에서, 같은 조회를 두 번 이상 수행하는 응용프로그램 프로세스에서는 추가 행을 볼 수 있습니다. 이러한 추가 행을 *팬텀 행(phantom row)*이라고 합니다.

예를 들면, 팬텀 행은 다음 조건에서 발생할 수 있습니다.

1. 응용프로그램 프로세스 P1은 검색 조건을 만족하는 *n* 행 집합을 읽어 들입니다.
2. 그런 다음 응용프로그램 프로세스 P2가 검색 조건을 만족하는 하나 이상의 행을 삽입한 다음 이 삽입설명을 확약합니다.
3. P1은 같은 검색 조건으로 다시 행 집합을 읽어 들이고, 원래의 행과 P2가 삽입한 행 모두를 확보하게 됩니다.

독점 잠금 외에도, 레벨 RS에서 수행 중인 응용프로그램 프로세스는 모든 규정 행에 최소한 공유 잠금을 확보합니다.

커서 안정성(CS, Cursor Stability)

RR 레벨과 같은 점은 다음과 같습니다.

- 커서 안정성(CS)은 다른 응용프로그램 프로세스가 변경한 행을 그 응용프로그램 프로세스가 확약할 때까지 읽을 수 없게 합니다.

RR 레벨과 다른 점은 다음과 같습니다.

- CS는 갱신 가능한 모든 커서의 현재 행이 다른 응용프로그램 프로세스에서 변경되지 않도록만 합니다. 그러므로, 작업 단위(UOW) 동안 읽혀진 행은 다른 응용프로그램 프로세스가 변경할 수 있습니다.

독점 잠금 외에도, 레벨 CS에서 수행 중인 응용프로그램 프로세스는 모든 커서의 현재 행에 대해 최소 공유 잠금을 확보합니다.

미확약 읽기(UR, Uncommitted Read)

읽기 전용 커서가 있는 SELECT INTO, FETCH, INSERT에 사용된 fullselect, UPDATE의 행 fullselect 및 스칼라 fullselect(어떤 곳에나 사용되는)의 경우, 레벨 UR은 다음을 허용합니다.

- 작업 단위(UOW)동안 읽은 행을 다른 응용프로그램 프로세스가 변경할 수 있습니다.
- 다른 응용프로그램 프로세스가 변경한 행을 비록 그 응용프로그램 프로세스가 변경사항을 확약하지 않더라도 읽을 수 있습니다.

다른 조작에 대해서는, 레벨 CS의 규칙이 적용됩니다.

분리 레벨 비교

4개의 분리 레벨에 대한 비교는 1449 페이지의 『부록I. 분리 레벨 비교』에서 찾을 수 있습니다.

분산 관계형 데이터베이스

*분산 관계형 데이터베이스(distributed relational database)*는 서로 다르지만 서로 연결되어 있는 컴퓨터 시스템간에 퍼져 있는 테이블과 기타 오브젝트의 세트로 구성됩니다. 각 컴퓨터 시스템에는 관계형 데이터베이스 관리 프로그램이 있어서 해당 환경에 있는 테이블을 관리합니다. 데이터베이스 관리 프로그램은 주어진 데이터베이스 관리 프로그램이 다른 컴퓨터 시스템에서 SQL문을 수행하는 것을 허용하는 방식으로 서로 통신하고 협력합니다.

분산 관계형 데이터베이스는 정규 리퀘스터-서버 프로토콜과 함수로 구축됩니다. *응용프로그램 리퀘스터*에서는 연결의 응용프로그램쪽 끝을 지원합니다. 이것은 응용프로그램에서의 데이터베이스 요청을 분산형 데이터베이스 네트워크에 적합한 통신 프로토콜로 변형시킵니다. 이러한 요청은 연결의 다른 끝에서 *응용프로그램 서버(AS)*에 의해 인식되고 처리됩니다. 함께 작업하면서, 응용프로그램 리퀘스터와 응용프로그램 서버(AS)는 통신과 위치 고려사항을 처리하여 응용프로그램이 이러한 고려사항과 상관없이 지역 데이터베이스처럼 액세스하도록 작업할 수 있습니다. 샘플 분산 관계형 데이터베이스 환경은 35 페이지의 그림3에서 보여주고 있습니다.

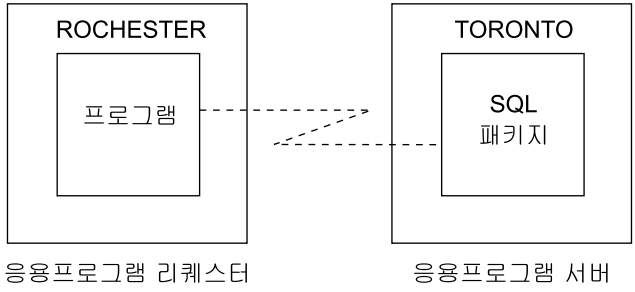


그림 3. 분산 관계형 데이터베이스 환경

DRDA 통신 프로토콜에 대해서는 *Distributed Relational Database Architecture Reference* SC26-4651을 참조하십시오.

응용프로그램 서버(AS)

응용프로그램 프로세스는 테이블이나 뷰를 참조하는 SQL문을 수행하기 전에 데이터베이스 관리 프로그램의 응용프로그램 서버(AS)에 연결되어야 합니다. CONNECT문은 응용프로그램 프로세스와 해당되는 서버 사이의 연결을 설정합니다.[9] 서버는 CONNECT문 실행시 변경될 수 있습니다.

응용프로그램 서버(AS)는 프로세스가 시작되는 환경에서 지역 또는 원격이 될 수 있습니다(분산 관계형 데이터베이스를 사용하지 않더라도 응용프로그램 서버(AS)는 존재합니다.). 이 환경에는 CONNECT문에서 식별할 수 있는 응용프로그램 서버(AS)를 설명하는 지역 디렉토리를 갖고 있습니다. 지역 디렉토리에 대해서는 *관리 안내서*에서 자세한 설명을 참조하십시오.

테이블이나 뷰를 참조하는 정적 SQL문을 수행하기 위해, 응용프로그램 서버(AS)는 명령문의 바인드된 양식을 사용합니다. 이 바인드된 명령문은 바인드 조작을 통해 데이터베이스 관리 프로그램이 이전에 작성한 패키지에서 가져온 것입니다.

대부분의 경우, 응용프로그램은 현재 응용프로그램이 연결되어 있는 응용프로그램 서버(AS)의 데이터베이스 관리 프로그램에서 지원하는 명령문과 절을 사용하는 데,

9. DB2 CLI 및 Embedded SQL은 다중 연결(각각 독립적인 트랜잭션임)을 지원하는 *동시 트랜잭션*이라고 하는 연결 모드를 지원합니다. 응용프로그램은 동일한 데이터베이스에 대해 여러 개의 동시적인 연결을 가질 수 있습니다. 여러 스레드 데이터베이스 액세스에 대한 세부사항은 *응용프로그램 개발 안내서*에서 자세한 내용을 참조하십시오.

응용프로그램이 이러한 명령문과 절을 지원하지 않는 데이터베이스 관리 프로그램의 응용프로그램 리퀘스터를 통해 수행되더라도 마찬가지입니다.

응용프로그램 서버(AS)를 사용하여 분산 데이터 소스의 시스템에서 조회를 제출하는 방법에 대해서는 50 페이지의 『서버 정의 및 서버 옵션』에서 자세한 내용을 참조하십시오.

CONNECT(유형 1) 및 CONNECT(유형 2)

두 가지 유형의 CONNECT문이 있습니다.

- CONNECT(유형 1)에서는 작업 단위(UOW)당 하나의 데이터베이스(원격 작업 단위(UOW)) 의미(semantics)를 제공합니다. 608 페이지의 『CONNECT(유형 1)』에서 자세한 내용을 참조하십시오.
- CONNECT(유형 2)는 작업 단위(UOW)당 여러 개의 데이터베이스(응용프로그램 직접 분산 작업 단위(DUOW)) 의미(semantics)를 지원합니다. 618 페이지의 『CONNECT(유형 2)』에서 자세한 내용을 참조하십시오.

원격 작업 단위

원격 작업 단위 기능은 SQL문의 원격 준비와 실행을 위해 제공됩니다. 컴퓨터 시스템 A에 있는 응용프로그램 프로세스는 컴퓨터 시스템 B에 있는 응용프로그램 서버(AS)에 연결될 수 있고, 하나 이상의 작업 단위(UOW) 내에서 B에 있는 오브젝트를 참조하는 정적 SQL이나 정적 SQL을 몇개든 수행할 수 있습니다. 응용프로그램 프로세스는 B에서 작업 단위(UOW)를 마친 후 컴퓨터 시스템 C 등등에 있는 응용프로그램 서버(AS)에 연결할 수 있습니다.

대부분의 SQL문은 다음과 같은 제한을 받으면서 원격으로 준비 및 수행할 수 있습니다.

- 하나의 SQL문에서 참조되는 모든 오브젝트는 같은 응용프로그램 서버(AS)가 관리해야 합니다.
- 한 작업 단위(UOW)에 있는 모든 SQL문은 같은 응용프로그램 서버(AS)에 의해 수행되어야 합니다.

원격 작업 단위 연결 관리

이 절에서는 응용프로그램 프로세스가 들어 갈 수 있는 연결 상태에 대해 개략적으로 설명합니다.

연결 상태:

응용프로그램 프로세스는 언제나 4가지 상태 중 하나에 있을 수 있습니다.

연결 가능하고 연결됨
연결 불가능하고 연결됨
연결 가능하고 연결되지 않음
내재 연결 가능(내재 연결이 사용 가능한 경우)

내재 연결이 사용 가능한 경우(39 페이지의 그림4 참조), 응용프로그램 프로세스는 처음에 *내재적으로 연결가능* 상태가 됩니다. 내재 연결을 사용할 수 없는 경우(40 페이지의 그림5 참조), 응용프로그램 프로세스는 처음에 *연결가능하고 연결되지 않음* 상태에 있게 됩니다.

내재 연결의 사용가능성은 설치 옵션, 환경 변수 및 인증 설정에 의해 결정됩니다. 설치시 내재적 연결 설정에 대한 *빠른 시작*에서 정보를 참조하고 환경 변수와 인증 설정에 대해 알려면 *관리 안내서*에서 정보를 참조하십시오.

내재 연결 가능 상태:

내재 연결을 사용할 수 있는 경우, 이것은 응용프로그램 프로세스의 초기 상태입니다. CONNECT RESET문을 사용하여 이 상태로 변화시킬 수 있습니다. 연결 불가능하고 연결됨 상태에서 COMMIT 또는 ROLLBACK문을 수행하고, 연결 가능하고 연결됨 상태에서 DISCONNECT문을 실행하면 이 상태로 될 수 있습니다.

연결 가능하고 연결됨 상태:

응용프로그램 프로세스가 응용프로그램 서버(AS)에 연결되었고, CONNECT문을 수행할 수 있습니다.

내재 연결을 사용할 수 있는 경우:

- 응용프로그램 프로세스는 CONNECT TO문이나 피연산자 없는 CONNECT문이 있는 연결 가능하고 연결되지 않음 상태에서 성공적으로 수행될 때 이 상태가 됩니다.

– 또한, 응용프로그램 프로세스는 CONNECT RESET, DISCONNECT, SET CONNECTION 또는 RELEASE 이외의 SQL문을 실행하는 경우에 내재적으로 연결가능 상태에서 이 상태로 들어갈 수도 있습니다.

내재 연결 사용 가능 여부에 관계없이, 다음 경우에 이 상태가 됩니다.

– CONNECT TO문이 연결 가능하고 연결되지 않음 상태에서 성공적으로 수행된 경우
– COMMIT나 ROLLBACK문이 성공적으로 수행되었거나 강제 구간 복원이 연결 불가능하고 연결됨 상태에서 발생한 경우

연결 불가능하고 연결됨 상태:

응용프로그램 프로세스가 응용프로그램 서버(AS)에 연결되었지만, CONNECT TO문이 응용프로그램 서버를 변경하기 위해 성공적으로 수행되지 않습니다. 프로세스는 CONNECT TO, 피연산자 없는 CONNECT, CONNECT RESET, DISCONNECT, SET CONNECTION, RELEASE, COMMIT 또는 ROLLBACK문 이외의 SQL문을 실행할 때 연결 가능하고 연결된 상태에서 이 상태가 됩니다.

연결 가능하고 연결되지 않음 상태:

응용프로그램 프로세스가 응용프로그램 서버(AS)에 연결되지 않습니다. 수행할 수 있는 유일한 SQL문은 CONNECT TO이며, 그렇지 않은 경우 오류(SQLSTATE 08003)가 발생합니다.

내재 연결 사용 가능 여부에 관계없이,

– CONNECT TO문이 발행될 때 오류가 발생하거나 작업 단위(UOW)에서 연결이나 구간복원을 잃게 하는 오류가 발생하는 경우 응용프로그램 프로세스는 이 상태가 됩니다. 응용프로그램 프로세스가 연결 가능 상태에 있지 않거나 서버 이름이 지역 디렉토리에 없음으로 인해 발생한 오류는 이 상태로 전이시키지 않습니다.

내재 연결을 사용할 수 없는 경우:

– CONNECT RESET 및 DISCONNECT문으로 인해 이 상태로 전이됩니다.

*상태 전이*가 다음 도표에 표시됩니다.

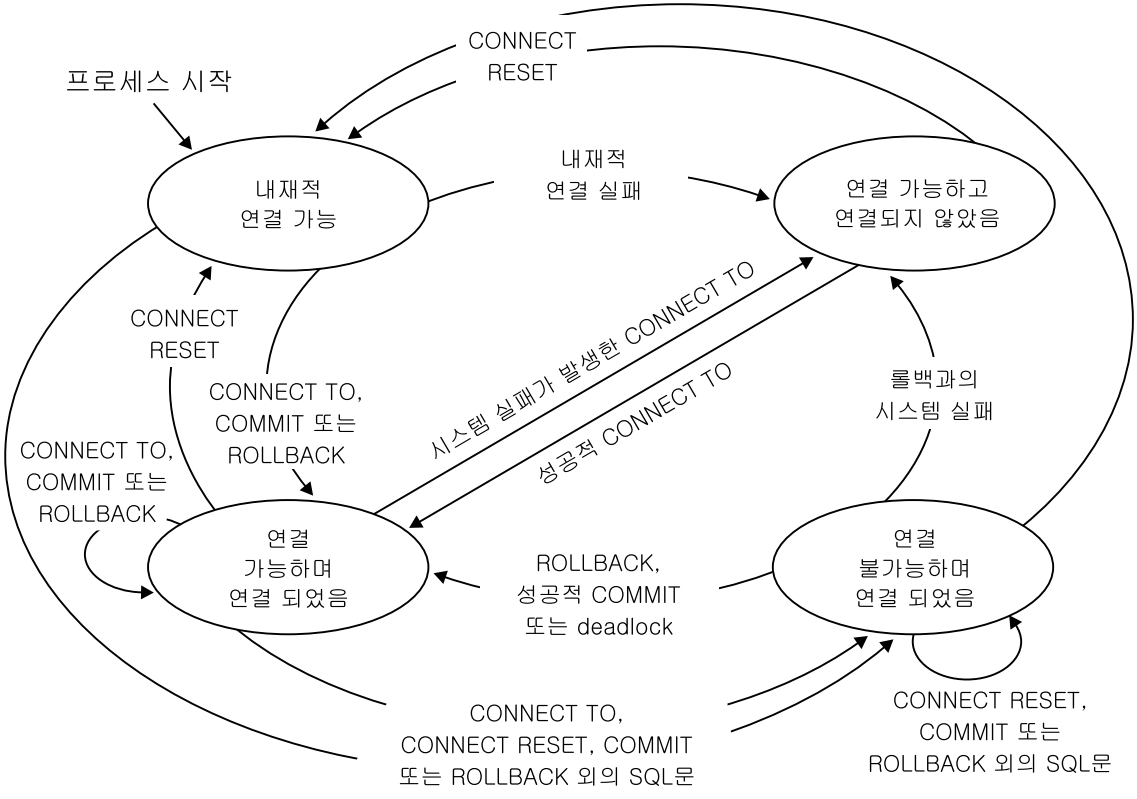


그림 4. 내재 연결을 사용할 수 있는 경우 연결 상태 전이

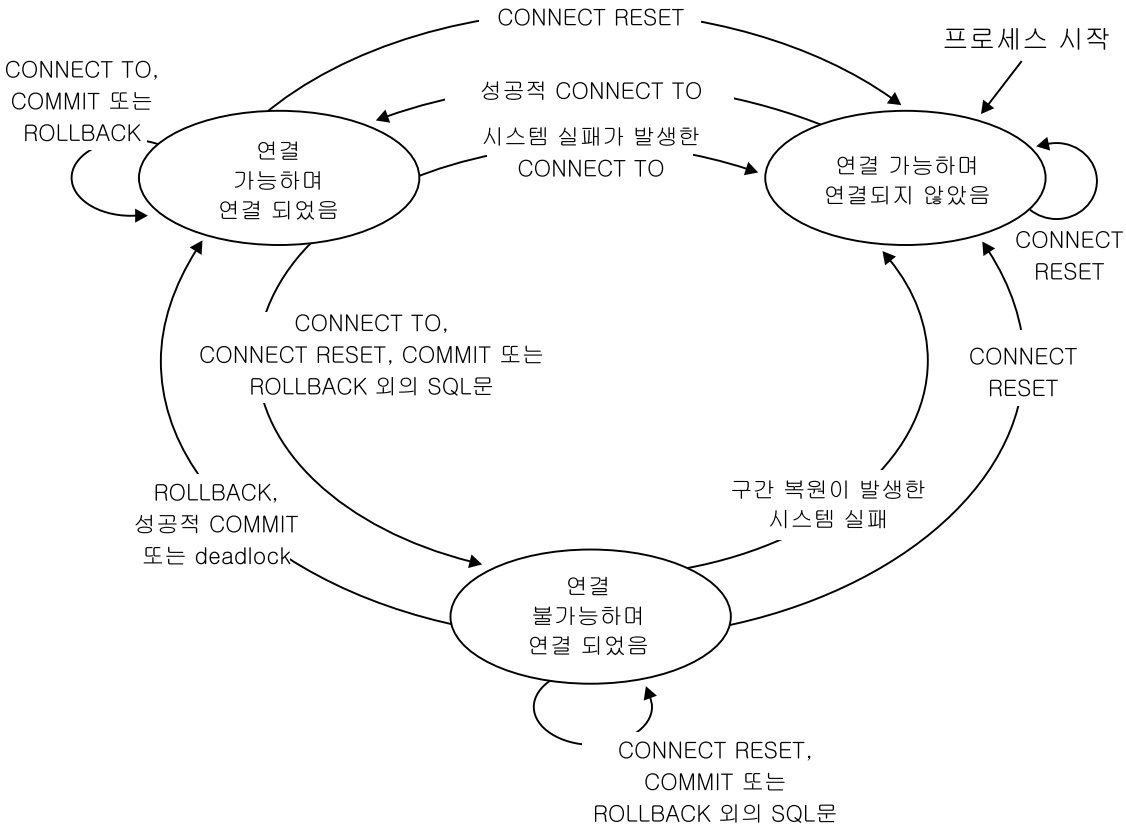


그림 5. 내재 연결을 사용할 수 없는 경우 연결 상태 전이

추가 규칙:

- CONNECT 자체가 연결 가능 상태에서 응용프로그램 프로세스를 제거하지 않으므로 연속적으로 CONNECT문을 수행하는 것은 오류가 아닙니다.
- 연속적으로 CONNECT RESET문을 수행하는 것은 오류입니다.
- CONNECT TO, CONNECT RESET, 피연산자 없는 CONNECT, SET CONNECTION, RELEASE, COMMIT 또는 ROLLBACK 이외의 SQL문을 수행한 다음 CONNECT TO문을 수행하는 것은 오류입니다. 오류를 피하려면, CONNECT RESET, DISCONNECT(COMMIT 또는 ROLLBACK문이 앞에 오는), COMMIT 또는 ROLLBACK문을 CONNECT TO를 수행하기 전에 수행해야 합니다.

응용프로그램 직접 분산 작업 단위

*응용프로그램 분산 작업 단위(DUOW) 기능*은 원격 작업 단위와 유사한 방법으로 SQL문의 원격 준비 및 실행을 위해 제공됩니다. 컴퓨터 시스템 A에 있는 응용프로그램 프로세스는 CONNECT나 SET CONNECTION문을 수행하여 시스템 B에 있는 응용프로그램 서버(AS)에 연결할 수 있습니다. 응용프로그램 프로세스는 작업 단위(UOW)를 종료하기 전에 B에서 오브젝트를 참조하는 여러 정적 SQL과 동적 SQL문을 수행할 수 있습니다. 하나의 SQL문에서 참조되는 모든 오브젝트는 같은 응용프로그램 서버(AS)가 관리해야 합니다. 그러나, 원격 작업 단위(UOW)와는 달리, 여러 응용프로그램 서버(AS)가 같은 작업 단위(UOW)에 포함될 수 있습니다. 확약이나 구간 복원 조작은 작업 단위(UOW)를 종료시킵니다.

응용프로그램 직접 분산 작업 단위 연결 관리

응용프로그램 직접 분산 작업 단위는 유형 2의 연결을 사용합니다. 유형 2 연결은 응용프로그램 프로세스를 식별된 응용프로그램 서버(AS)로 연결하여, 응용프로그램 직접 분산 작업 단위에 대한 규칙을 세웁니다.

응용프로그램 프로세스와 연결 상태에 대한 개요

유형 2 응용프로그램 프로세스는

- 항상 연결 가능합니다.
- *연결됨* 상태나 *연결되지 않음* 상태에 있습니다.
- 0 또는 그 이상의 연결을 갖습니다.

응용프로그램 프로세스의 각 연결은 연결에 대한 응용프로그램 서버의 데이터베이스 별명으로 고유하게 식별됩니다.

항상 각 연결은 다음 연결 상태 집합 중 하나를 갖습니다.

- *현재* 및 *보류*
- *현재* 및 *해제-보류*
- *휴면* 및 *보류*
- *휴면* 및 *해제-보류*

초기 상태 및 상태 전이: 유형 2 응용프로그램 프로세스는 처음에 *연결되지 않음 상태*이므로 연결을 갖고 있지 않습니다.

연결은 일차적으로 *현재* 및 *보류 상태*에 있습니다.

다음 도표는 상태 전이를 보여 줍니다.

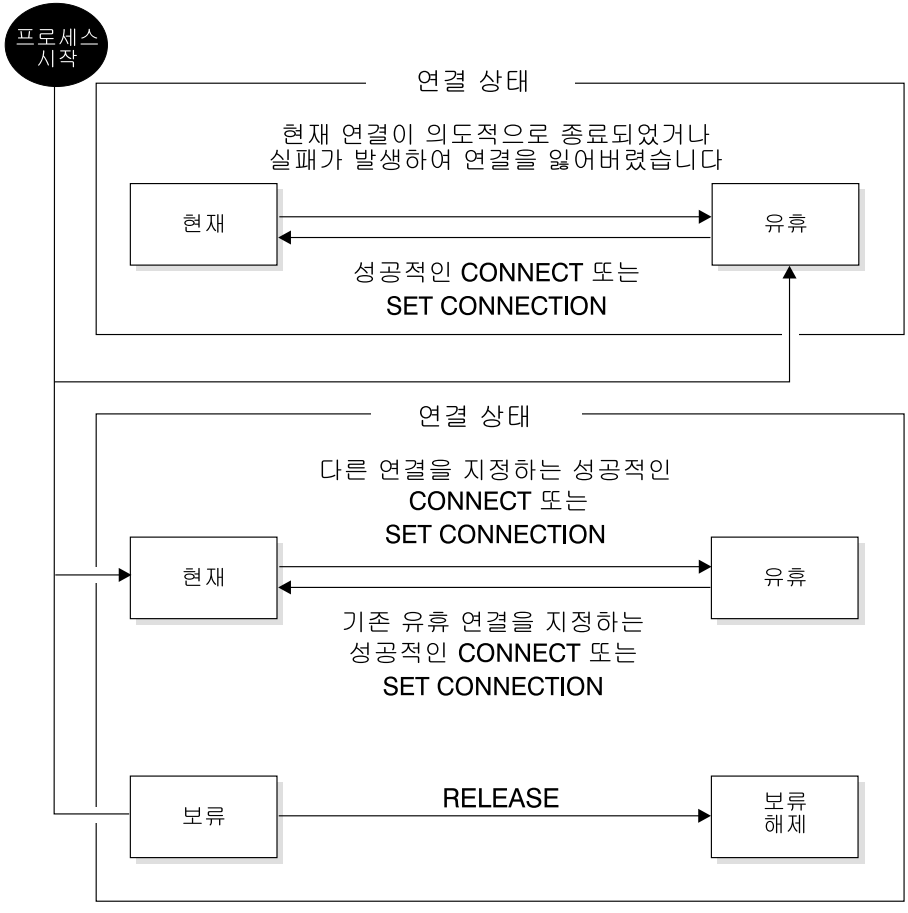


그림 6. 응용프로그램 직접 분산 작업 단위 연결 및 응용프로그램 프로세스 연결 상태 전이

응용프로그램 프로세스 연결 상태: 다른 응용프로그램 서버(AS)는 CONNECT 문을 명확하게 또는 내재적으로 수행하여 구축할 수 있습니다. [10] 다음 규칙이 적용됩니다.

10. 유형 2 내재 연결은 유형 1보다 제한적임을 기억하십시오. 618 페이지의 『CONNECT(유형 2)』에서 자세한 내용을 참조하십시오.

- 컨텍스트는 동일한 응용프로그램 서버(AS)에 동시에 두 개 이상의 연결을 가질 수 없습니다. 동일한 DB2 Universal Database에 대해 동시에 여러 개의 연결을 지원하는 데 대한 *관리 안내서* 및 *응용프로그램 개발 안내서*에서 정보를 참조하십시오.
- 응용프로그램 프로세스가 SET CONNECTION문을 수행할 때, 지정된 위치 이름은 응용프로그램 프로세스의 연결 세트에 있는 기존의 연결이어야 합니다.
- 응용프로그램 프로세스가 CONNECT문을 수행하고, SQLRULES(STD) 옵션이 유효한 경우, 지정된 서버 이름은 응용프로그램 프로세스의 연결 집합에 있는 기존 연결이 아니어야 합니다. SQLRULES 옵션에 대해 45 페이지의 『분산 작업 단위(DUOW) 의미 정의 옵션』에서 자세한 내용을 참조하십시오.

응용프로그램 프로세스에 현재 연결이 있는 경우, 응용프로그램 프로세스는 *연결됨* 상태에 있게 됩니다. CURRENT SERVER 특수 레지스터에는 현재 연결 응용프로그램 서버(AS) 이름이 들어 있습니다. 응용프로그램 프로세스는 응용프로그램 서버(AS)가 관리하는 오브젝트를 참조하는 SQL문을 수행할 수 있습니다.

연결되지 않음 상태에 있는 응용프로그램 프로세스가 CONNECT나 SET CONNECTION문을 성공적으로 수행하는 경우, 연결됨 상태가 됩니다. 응용프로그램에 연결은 없지만 SQL문이 발행되면, DB2DBDFT 환경 변수에서 제공하는 내재 연결이 기본 데이터베이스에 정의됩니다.

응용프로그램 프로세스에 현재 연결이 없는 경우, 응용프로그램 프로세스는 *연결되지 않음* 상태에 있게 됩니다. 수행할 수 있는 유일한 SQL문은 CONNECT, DISCONNECT ALL, 데이터베이스를 지정하는 DISCONNECT, SET CONNECTION, RELEASE, COMMIT 및 ROLLBACK입니다.

현재 연결을 고의적으로 종료했거나 응용프로그램 서버에서 구간 복원 작업의 실패와 연결 손실로 인해 SQL문 수행에 실패한 경우, *연결됨 상태*에 있던 응용프로그램 프로세스는 *연결되지 않음* 상태가 됩니다. DISCONNECT문을 정상적으로 수행하거나 연결이 *해제-보류 상태*로 있을 때 확약 조작을 정상적으로 수행하여, 연결을 의도적으로 종료합니다. DISCONNECT 사전 처리 컴파일러 옵션에 지정된 다른 옵션이 연결 종료에 영향을 미칩니다. AUTOMATIC으로 설정된 경우, 모든 연결이 종료됩니다. CONDITIONAL로 설정된 경우, WITH HOLD로 지정된 커서가 열려 있지 않은 모든 연결이 종료됩니다.

연결 상태: 응용프로그램 프로세스가 CONNECT문을 수행하고 서버 이름이 응용프로그램 리퀘스터에 알려졌지만 응용프로그램 프로세스의 기존 연결 집합에 없는 경우,

- 현재 연결은 *휴면 상태*에 들어가고,
- 서버 이름은 연결 집합에 추가되며,
- 새로운 연결이 *현재 상태*와 *보류 상태*에 있게 됩니다.

서버 이름이 이미 응용프로그램 프로세스의 기존 연결 집합에 있고, 응용프로그램이 옵션 SQLRULES(STD)로 사전 처리 컴파일된 경우, 오류(SQLSTATE 08002)가 발생합니다.

- ***보류 및 해제 보류 상태:*** RELEASE문은 연결이 보류 상태인지 또는 해제 보류 상태인지를 제어합니다. *해제-보류 상태*란 다음번 성공적인 확약 조작시 연결이 끊어짐을 의미합니다(구간복원은 연결에 영향을 미치지 않습니다.). *보류 상태*는 다음 조작시 연결이 끊어지지 않습니다. 모든 연결은 *초기*에 보류 상태에 있고, RELEASE문을 사용하여 해제 -보류 상태로 이동됩니다. 일단 해제-보류 상태에 있게 되면, 연결을 다시 보류 상태로 되돌릴 수 없습니다. 연결은 ROLLBACK문을 발행했거나 확약 동작이 성공하지 못해 구간복원이 발생한 경우 작업 단위(UOW) 내에서 해제-보류 상태에 있게 됩니다.

  연결이 해제되도록 명확하게 표시되지 않더라도, 확약 조작이 DISCONNECT 사전 처리 컴파일(precompiler) 옵션의 조건을 만족하는 경우 확약 조작에 의해 연결이 끊어질 수 있습니다.
- ***현재 및 휴면 상태:*** 연결 상태가 *보류*인지 또는 *해제 보류*인지에 관계없이, 연결을 *현재 상태* 또는 *휴면 상태*가 되게 할 수 있습니다. *현재 상태*는 연결이 이 상태에 있는 동안 수행하는 SQL문에 사용되는 것임을 의미합니다. *휴면 상태*는 연결이 현재 상태가 아님을 의미합니다. 휴면 연결에서 수행될 수 있는 유일한 SQL문은 COMMIT와 ROLLBACK이거나, ALL(모든 연결의 경우) 또는 특정 데이터베이스 이름을 지정할 수 있는 DISCONNECT 및 RELEASE입니다. SET CONNECTION과 CONNECT문은 기존의 연결이 *휴면 상태*에 있는 동안 명명 서버의 연결을 *현재 상태*로 변경시킵니다. 언제나, 하나의 연결만이 *현재 상태*에 있게 됩니다. 휴면 연결이 같은 작업 단위(UOW)에서 현재 상태가 되면, 모든 잠금, 커서 및 준비된 명령문은 같은 상태가 되고, 연결이 현재 상태인 경우는 그 마지막 사용에 영향을 미칩니다.

연결 종료시: 연결이 종료될 경우, 연결을 통해 응용프로그램 프로세스에 의해 확보된 모든 자원과, 연결을 작성하고 유지보수하기 위해 사용된 모든 자원의 할당이 취소됩니다. 예를 들면, 응용프로그램 프로세스가 RELEASE문을 수행하면, 열린 커서는 연결이 다음 확약 조작 중에 종료될 때 닫힙니다.

연결은 또한 통신에 실패한 경우에도 종료될 수 있습니다. 종료된 연결이 현재 연결이었을 경우 응용프로그램 프로세스는 연결되지 않음 상태로 됩니다.

응용프로그램 프로세스의 모든 연결은 프로세스가 끝나면 종료됩니다.

분산 작업 단위(DUOW) 의미 정의 옵션

유형 2 연결의 의미는 사전 처리 컴파일러 옵션 집합으로 결정됩니다. 이것은 굵은체와 밑줄 친 텍스트로 표시된 기본값과 함께 아래에 간략히 요약되어 있습니다. *Command Reference* 또는 *Administrative API Reference*에서 자세히 매뉴얼을 참조하십시오.

- CONNECT(**1** | 2)

  CONNECT문이 유형 1이나 유형2 중 어느 것으로 처리될 것인지를 결정합니다.

- SQLRULES(**DB2** | STD)

  유형 2 CONNECT가, CONNECT가 휴면 연결로 전환될 수 있는 DB2 규칙에 따라 처리되는지 또는 이것을 허용하지 않는 SQL92 표준(STD) 규칙에 따라 처리되는지를 결정합니다.

- DISCONNECT(**EXPLICIT** | CONDITIONAL | AUTOMATIC)

  확약 조작이 발생할 때 연결이 해제되는 데이터베이스 연결을 지정합니다. 즉, 다음과 같습니다.

  – SQL RELEASE문에 의해 해제하도록 명시적으로 표시된 연결(EXPLICIT) 또는

  – 열려 있는 WITH HOLD 커서가 없는 연결과 해제로 표시된 연결(CONDITIONAL) 또는 [11]

  – 모든 연결(AUTOMATIC).

11. CONDITIONAL 옵션은 버전 2 이전의 하위 레벨 서버에서는 제대로 작동하지 않습니다. 이 경우 WITH HOLD 커서의 유무에 관계없이 연결이 끊어집니다.

- SYNCPOINT(**ONEPHASE** | TWOPHASE | NONE)

  확약이나 구간 복원이 여러 데이터베이스 연결 사이에서 어떻게 조정되는지를 지정합니다.

  ONEPHASE 갱신은 작업 단위 내의 한 데이터베이스에서만 발생할 수 있으며, 모든 다른 데이터베이스는 읽기 전용입니다. 다른 데이터베이스를 갱신하려하면, 오류(SQLSTATE 25000)가 발생합니다.

  TWOPHASE 런타임 트랜잭션 관리 프로그램(TM)을 사용하여 이 프로토콜을 지원하는 데이터베이스들 사이에서 2단계 확약을 조정합니다.

  NONE 2단계 확약을 수행하기 위해 하나의 갱신자, 여러 리더를 강제로 수행하지 않습니다. COMMIT나 ROLLBACK문이 수행되면, 개개의 COMMIT나 ROLLBACK이 모든 데이터베이스에 부과됩니다. 하나 이상의 구간 복원에 실패하면, 오류(SQLSTATE 58005)가 발생합니다. 하나 이상의 확약에 실패하면, 오류(SQLSTATE 40003)가 발생합니다.

위의 옵션 런타임 특수 SET CLIENT API를 사용하여 대체할 수 있습니다. 이들의 현재 설정값은 특수 QUERY CLIENT 응용프로그램 인터페이스(API)를 사용하여 확보할 수 있습니다. 이들은 SQL문이 아닙니다. 이들은 다양한 호스트 언어와 명령행 처리기에 정의된 API임을 기억하십시오. 이러한 API는 *Command Reference* 와 *Administrative API Reference* 매뉴얼에 정의되어 있습니다.

데이터 표현에 관한 고려사항

다른 시스템에서는 다른 방법으로 데이터를 표시합니다. 데이터가 한 시스템에서 다른 시스템으로 이동할 때, 때때로 데이터 변환을 수행해야 합니다. DRDA를 지원하는 제품은 자동으로 수신하는 시스템에서 필요한 변환을 수행합니다. 숫자 데이터에서, 변환에 필요한 정보는 데이터의 유형과 전송하는 시스템에서 데이터 유형이 표현되는 방법입니다. 문자 데이터에서는, 문자열을 변환시키기 위해 추가 정보가 필요합니다. 문자열 변환은 데이터의 코드 페이지와 그 데이터와 수행될 조작 모두에 따라 달라집니다. 문자 변환은 IBM 문자 데이터 표현 아키텍처(CRDA)에 따라 수행됩니다. 문자 변환에 대한 자세한 내용은 *Character Data Representation Architecture Reference* SC09-1390를 참조하십시오.

DB2 연합 시스템

이 섹션은 DB2 연합 시스템 요소의 개요, 시스템의 관리자 및 사용자가 수행하는 타스크의 개요 및 이들 타스크와 연관된 개념에 대한 설명을 제공합니다.

연합 서버, 연합 데이터베이스 및 데이터 소스

*DB2 연합 시스템*은 다음으로 구성된 분산 컴퓨팅 시스템입니다.

- *연합 서버*라고 하는 DB2 서버.

  DB2 설치시, 몇 개의 DB2 인스턴스든 연합 서버로 기능하도록 구성될 수 있습니다.

- 연합 서버가 조회를 보내는 여러 개의 데이터 소스.

  각 데이터 소스는 관계형 데이터베이스 관리 시스템의 인스턴스에 더하여 인스턴스가 지원하는 데이터베이스들로 구성됩니다. DB2 연합 시스템에서 데이터 소스는 Oracle 인스턴스 및 DB2 계열 구성원의 인스턴스를 포함할 수 있습니다.

  데이터 소스는 반자동일 수 있습니다. 예를 들어, 연합 서버는 Oracle 응용프로그램이 이들 데이터 소스를 액세스할 수 있는 것과 동시에 Oracle 데이터 소스로 조회를 보낼 수 있습니다. DB2 연합 시스템은 Oracle이나 기타 데이터 소스에 대한 액세스를 독점하거나 제한할 수 없습니다(무결성 및 잠금 제한조건을 넘어서).

일반 사용자 및 클라이언트 응용프로그램에게 데이터 소스는 하나의 집합적 데이터베이스로 나타납니다. 실제로, 연합 서버 내에 있는 *연합 데이터베이스*라고 하는 데이터베이스와의 사용자 및 응용프로그램 인터페이스. 데이터 소스로부터 데이터를 확보하기 위해 이들은 DB2 SQL에서 연합 데이터베이스로 조회를 제출합니다. 그런 후 DB2는 조회를 적절한 데이터 소스로 분산합니다. DB2는 또한 조회 최적화를 위한 액세스 플랜을 제공합니다(일부 경우에는 이러한 플랜이 데이터 소스가 아니라 연합 서버에서의 조회 처리를 호출합니다.). 마지막으로, DB2는 요청된 데이터를 수집하여 이를 사용자 및 응용프로그램에 전달합니다.

연합 서버로부터 데이터 소스에 제출된 조회는 읽기 전용이어야 합니다. 데이터 소스에 쓰려면(예를 들어, 데이터 소스 테이블을 갱신하기 위해) 사용자 및 응용프로그램이 *통과* 라고 하는 특별한 모드에서 데이터 소스 자체의 SQL을 사용해야 합니다.

DB2 연합 시스템에서 수행하는 타스크

이 절에서는 연합 시스템을 형성하고 사용하기 위해 사용자가 수행하는 타스크와 연관된 개념들을 소개합니다(이 절과 그 다음에 나오는 절들에서, 사용자라는 용어는 연합 시스템에 대해 작업하는 모든 유형의 개인을 말합니다. 예를 들어, 데이터베이스 관리자, 응용프로그램 프로그래머, 일반 사용자 등이 있습니다).

다음의 타스크 목록은 전형적으로 타스크를 수행하는 사용자들의 유형을 식별합니다. 사용자들의 다른 유형 역시 이들 타스크를 수행할 수 있다는 것을 유념하십시오. 예를 들어, 목록에서 DBA가 전형적으로 연합 시스템을 액세스하기 위한 권한 부여와 데이터 소스를 액세스하기 위한 권한 사이의 맵핑을 작성한다고 알려줍니다. 그러나 응용프로그램 프로그래머와 일반 사용자 역시 이 타스크를 수행할 수 있습니다.

DB2 연합 시스템을 형성하고 사용하려면:

1. DBA는 DB2 서버를 연합 서버로 지정합니다. 이 방법에 대한 *설치 및 구성 보충 설명서*에서 자세한 정보를 참조하십시오.
2. 액세스를 위한 데이터 소스가 설정됩니다.
   a. DBA가 연합 데이터베이스에 연결합니다.
   b. DBA가 연합 시스템에 포함될 데이터 소스의 각 카테고리에 대한 랩퍼를 작성합니다(*랩퍼*는 연합 서버가 데이터 소스와 상호 작용하는 메카니즘입니다. 50 페이지의 『랩퍼 및 랩퍼 모듈』에서 자세한 내용을 참조하십시오.).
   c. DBA는 연합 서버에 각 데이터 소스에 대한 설명을 제공합니다(설명을 *서버 정의*라 합니다. 50 페이지의 『서버 정의 및 서버 옵션』에서 자세한 내용을 참조하십시오.).
   d. 연합 데이터베이스에 액세스하기 위해 사용되는 사용자의 권한 부여 ID가 데이터 소스에 액세스하기 위해 사용되는 사용자의 권한 부여 ID와 다를 경우, DBA는 두 권한 부여 ID 사이의 연관을 정의합니다(이 연관을 *사용자 맵핑*이라 합니다. 53 페이지의 『사용자 맵핑 및 사용자 옵션』에서 자세한 내용을 참조하십시오.).
   e. DB2 데이터 유형과 데이터 소스 데이터 유형 사이의 기본 맵핑이 사용자 요구사항과 일치하지 않을 경우, DBA는 필요에 따라 맵핑을 수정합니다(*데이터 유형 맵핑*은 두 개의 호환 가능 데이터 유형 사이의 정의된 연합

입니다. 하나는 연합 데이터베이스에 의해, 또 하나는 데이터 소스에 의해 지원됩니다. 54 페이지의 『데이터 유형 맵핑』에서 자세한 내용을 참조하십시오.).

f. DB2 함수와 데이터 소스 함수 사이의 기본 맵핑이 사용자 요구사항과 일치하지 않을 경우, DBA는 필요에 따라 맵핑을 수정합니다(*함수 맵핑*은 두 개의 호환 가능 함수 사이의 정의된 연합입니다. 하나는 연합 데이터베이스에 의해, 또 하나는 데이터 소스에 의해 지원됩니다. 55 페이지의 『함수 맵핑, 함수 템플리트 및 함수 맵핑 옵션』에서 자세한 내용을 참조하십시오.).

g. DBA 및 응용프로그램 프로그래머는 액세스될 데이터 소스 테이블 및 뷰에 대한 별명을 작성합니다(*별명*은 연합 시스템이 데이터 소스 테이블이나 뷰를 참조하는 식별자입니다. 56 페이지의 『별명(Nickname) 및 컬럼 옵션』에서 자세한 내용을 참조하십시오.).

h. 선택적: 데이터 소스 테이블에 색인이 없으면, DBA가 연합 서버에 실제 색인의 정의가 포함할 동일한 종류의 정보를 제공할 수 있습니다. 데이터 소스 테이블에 연합 서버가 알지 못하는 색인이 있으면, DBA가 서버에게 색인의 존재에 대해 알릴 수 있습니다. 어느 경우에든, DBA가 제공하는 정보는 DB2가 테이블 데이터의 조회 최적화를 돕습니다(이 정보를 *색인 스펙*이라 부릅니다. 57 페이지의 『색인 스펙』에서 자세한 내용을 참조하십시오.).

3. 응용프로그램 프로그래머 및 일반 사용자는 데이터 소스로부터 정보를 검색합니다.

   • DB2 SQL을 사용할 경우, 응용프로그램 프로그래머와 일반 사용자는 별명에 의해 참조되는 테이블과 뷰를 조회합니다. (두 개 이상의 데이터 소스로 지정된 조회를 *분산 요청*이라고 합니다. 58 페이지의 『분산 요청』에서 자세한 내용을 참조하십시오.)

   조회 처리시, 연합 서버는 DB2 SQL에 의해서는 지원되나 데이터 소스의 SQL에 의해서는 지원되지 않는 조작을 수행할 수 있습니다. (이 능력을 *보충*이라고 합니다. 59 페이지의 『보충』에서 자세한 내용을 참조하십시오.)

   • 응용프로그램 프로그래머 및 일반 사용자는 종종 데이터 소스 자체의 SQL에서 데이터 소스로 조회, DML문 및 DDL문을 제출할 수도 있습니다. 프

로그래머와 사용자는 통과 모드에서 이를 수행할 수 있습니다. (60 페이지의 『통과』에서 자세한 내용을 참조하십시오.)

다음 절에서는 이 타스크 목록에서 언급된 개념들을 목록에 나타나는 순서대로 설명합니다. 이들 절의 일부는 관련 개념도 소개합니다.

랩퍼 및 랩퍼 모듈

랩퍼는 연합 서버가 데이터 소스와 통신하여 데이터를 검색하는 메카니즘입니다. 랩퍼를 구현하기 위해, 서버는 *랩퍼 모듈*이라고 하는 라이브러리에 보관된 루틴을 사용합니다. 이들 루틴은 서버로 하여금 데이터 소스에 연결하고 이로부터 반복적으로 데이터를 검색하는 것과 같은 조작을 수행할 수 있게 합니다.

세가지 랩퍼가 있습니다.

- DRDA의 기본 이름을 가진 랩퍼는 모든 DB2 계열 데이터 소스에 사용됩니다.
- SQLNET의 기본 이름을 가진 랩퍼는 Oracle의 SQL*Net 소프트웨어에 의해 지원되는 모든 Oracle 데이터 소스에 사용됩니다.
- NET8의 기본 이름을 가진 랩퍼는 Oracle의 Net8 소프트웨어에 의해 지원되는 모든 Oracle 데이터 소스에 사용됩니다.

랩퍼는 CREATE WRAPPER문으로 연합 서버에 등록됩니다. 949 페이지의 『CREATE WRAPPER』에서 자세한 내용을 참조하십시오.

서버 정의 및 서버 옵션

DBA가 연합 서버로 하여금 데이터 소스와 상호 작용하게 하는 랩퍼를 등록한 후, DBA는 그러한 데이터 소스를 연합 데이터베이스에 정의합니다. 이 절에서는,

- DBA가 제공하는 정의를 설명합니다.
- 서버 정의의 부분을 포함하는 *서버 옵션*이라고 하는 특정 매개변수 지정을 위한 SQL을 설명합니다.
- 『서버』라는 용어의 서로 다른 의미를 구별합니다.

서버 정의에 대한 소개

연합 데이터베이스로 데이터 소스를 정의할 때, DBA는 데이터 소스에 속하는 정보뿐만 아니라 데이터 소스에 대한 이름도 제공합니다. 이 정보에는 데이터 소스가 인스턴스인 RDBMS의 유형과 버전, 그리고 데이터 소스에 대한 RDBMS의 이

름이 포함됩니다. 여기에는 RDBMS 특유의 메타데이터도 포함됩니다. 예를 들어, DB2 계열 데이터 소스에는 여러 개의 데이터베이스가 있을 수 있으며, 그러한 데이터 소스의 정의는 연합 서버가 어느 데이터베이스에 연결할 수 있는지 지정해야 합니다. 반대로, Oracle 데이터 소스에는 하나의 데이터베이스가 있으며, 연합 서버는 그 이름을 알 필요없이 데이터베이스에 연결할 수 있습니다. 그러므로, 이름은 데이터 소스의 연합 서버 정의에 포함되지 않습니다.

DBA가 제공하는 이름과 정보를 모아서 서버 정의라고 합니다. 이 용어는 데이터 소스가 데이터 요청에 응답하며 따라서 당연히 서버라는 사실을 반영합니다. 다른 용어들 역시 이러한 사실을 반영합니다. 예를 들면,

- 서버 정의 내의 일부 정보는 서버 옵션으로 저장됩니다. 그러므로, 데이터 소스의 이름은 NODE라고 하는 서버 옵션의 값으로 저장됩니다. DB2 계열 데이터 소스의 경우, 연합 서버가 연결하는 데이터베이스의 이름 DBNAME이라고 하는 서버 옵션의 값으로서 보관됩니다.
- 서버 정의의 작성 및 수정을 위한 SQL문은 각각 CREATE SERVER와 ALTER SERVER라고 합니다.

서버 옵션 설정을 위한 SQL문

CREATE SERVER, ALTER SERVER 및 SET SERVER OPTION문을 통해 값들이 서버 옵션에 지정됩니다.

CREATE SERVER 및 ALTER SERVER문은 서버 옵션을 데이터 소스로의 연속적인 연결 동안 지속되는 값들에 설정합니다. 이들 값들은 카탈로그에 보관됩니다. 다음 상황을 고려해 보십시오. 연합 시스템 DBA가 CREATE SERVER문을 사용하여 새로운 Oracle 데이터 소스를 연합 시스템에 정의합니다. 이 데이터 소스의 데이터베이스는 연합 데이터베이스가 사용하는 것과 동일한 조합 순서를 사용합니다. DBA는 최적화 알고리즘이 이 일치를 알고 있어 이를 성능 촉진에 이용할 수 있기를 원합니다. 이에 따라, CREATE SERVER문에서 DBA는 COLLATING_SEQUENCE라는 서버 옵션을 'Y'(데이터 소스와 연합 데이터베이스에서의 조합 순서가 동일함을 의미하는 yes)로 설정합니다. 'Y' 설정이 카탈로그에 기록되고, 이는 사용자와 응용프로그램이 Oracle 데이터 소스에 액세스하는 동안 유효하게 유지됩니다.

몇달 후, Oracle DBA가 Oracle 데이터 소스의 조합 순서를 변경합니다. 그러므로, 연합 시스템 DBA가 COLLATING_SEQUENCE를 'N'(데이터 소스의 조합 순서가 연합 데이터베이스와 동일하지 않음을 의미하는 no)으로 재설정합니다. DBA는 ALTER SERVER문을 사용하여 이러한 갱신을 합니다. 사용자와 응용프로그램이 데이터 소스에 계속 액세스함에 따라 카탈로그도 갱신되고, 새 설정이 적용됩니다.

SET SERVER OPTION문은 연합 데이터베이스에 대한 단일 연결 지속 시간 동안 서버 옵션 값을 임시로 겹쳐씁니다. 겹쳐쓰는 값은 카탈로그에 보관되지 않습니다.

PLAN_HINTS라는 서버 옵션은 DB2가 Oracle 데이터 소스에 플랜 힌트라고 하는 명령문 프레그먼트를 제공할 수 있게 하는 값으로 설정될 수 있습니다. 플랜 힌트는 Oracle 최적화 알고리즘이 작업을 할 수 있게 돕습니다. 예를 들어, 플랜 힌트든 최적화 알고리즘이 테이블 액세스에 어느 색인을 사용할지 또는 결과 집합을 위한 데이터 검색에 어느 테이블 조인 순서를 사용할지 결정하는 것을 도울 수 있습니다.

데이터 소스 ORACLE1 및 ORACLE2의 경우, PLAN_HINTS 서버 옵션은 기본값인 'N'(플랜 힌트로 이러한 데이터 소스를 공급하지 않음)으로 설정됩니다. 그런 후 프로그래머는 ORACLE1 및 ORACLE2로부터의 데이터에 대한 분산 요청을 작성하며, 프로그래머는 플랜 힌트가 이들 데이터 소스에 있는 최적화 알고리즘이 이 데이터 액세스를 위한 전략을 향상시키는 것을 도울 것입니다. 이에 따라, 프로그래머는 SET SERVER OPTION문을 사용하여 'N'을 'Y'(플랜 힌트를 제공하라는 의미인 yes)로 겹쳐씁니다. 요청을 포함하는 응용프로그램이 연합 데이터베이스에 연결되어 있는 동안에만 'Y'가 계속 영향을 미칩니다. 카탈로그에 보관되지 않습니다.

795 페이지의 『CREATE SERVER』, 519 페이지의 『ALTER SERVER』 및 1180 페이지의 『SET SERVER OPTION』에서 자세한 정보를 참조하십시오. 모든 서버 옵션 및 그 설정값에 대한 1407 페이지의 『서버 옵션』에서 설명을 참조하십시오.

"서버"의 세가지 의미

서버 정의 및 *서버 옵션*이라는 용어와, 이전 절에서 논의된 SQL문에서 *서버*라는 단어는 데이터 소스만을 지칭합니다. 연합 서버나 DB2 응용프로그램 서버(AS)는 지칭하지 않습니다.

그러나 DB2 응용프로그램 서버 및 연합 서버의 개념은 겹쳐집니다. 34 페이지의 『분산 관계형 데이터베이스』에서 지시된 대로, 응용프로그램 서버는 응용프로그램 프로세스가 연결하여 요청을 제출하는 데이터베이스 관리 프로그램 인스턴스입니다. 이는 또한 연합 서버에도 적용되므로, 연합 서버는 응용프로그램 서버(AS)의 유형입니다. 그러나 다음의 두가지가 이를 다른 응용프로그램 서버(AS)와 구별합니다.

- 궁극적으로 데이터 소스를 위한 것인 요청을 검색하도록 구성되며, 이는 이들 요청을 데이터 소스에 분산합니다.
- 다른 응용프로그램 서버(AS)처럼, 연합 서버는 DRDA 통신 프로토콜을 사용하여 DB2 계열 인스턴스와 통신합니다. 다른 응용프로그램 서버(AS)와는 달리, 연합 서버는 sqlnet 및 net8 통신 프로토콜을 사용하여 Oracle 인스턴스와 통신합니다.

사용자 맵핑 및 사용자 옵션

연합 서버는 허가된 사용자 또는 응용프로그램의 분산 요청을 다음 조건하에서 데이터 소스에 전송할 수 있습니다.

- 사용자나 응용프로그램이 연합 데이터베이스와 데이터 소스 둘다에 대해 동일한 사용자 ID를 사용합니다. 또한, 데이터 소스가 암호를 필요로 하면 사용자나 응용프로그램이 연합 데이터베이스와 데이터 소스에 대해 동일한 암호를 사용합니다.
- 연합 데이터베이스를 액세스하기 위한 사용자나 응용프로그램의 권한 부여는 데이터 소스를 액세스하기 위한 사용자나 응용프로그램의 권한 부여와 다소 다릅니다. 또한, 사용자나 응용프로그램 요청이 데이터 소스를 액세스할 때, 액세스가 권한 부여될 수 있도록 연합 데이터베이스 권한 부여가 데이터 소스 권한 부여로 바뀝니다. 이러한 변경은 사용자 맵핑이라고 하는 정의된 연관이 두 권한 사이에 존재할 경우에만 발생합니다.

사용자 맵핑은 CREATE USER MAPPING 및 ALTER USER MAPPING문으로 정의되고 수정될 수 있습니다. 이들 명령문은 권한 부여와 관련된 값들이 지

정되는 *사용자 옵션*이라고 하는 매개변수를 포함합니다. 예를 들어, 사용자가 연합 데이터베이스 및 데이터 소스에 대해 동일한 ID를 가지고 있으나, 다른 암호를 가지고 있다고 가정합시다. 사용자가 데이터 소스를 액세스하기 위해서는, 암호를 서로에게 맵핑해야 합니다. 이는 데이터 소스에 있는 암호가 REMOTE_PASSWORD라고 하는 사용자 옵션에 값으로 지정되는 CREATE USER MAPPING문에서 수행될 수 있습니다.

928 페이지의 『CREATE USER MAPPING』, 569 페이지의 『ALTER USER MAPPING』 및 *관리 안내서*에서 자세한 정보를 참조하십시오. 사용자 옵션 및 그 설정값에 대한 1412 페이지의 『사용자 옵션』에서 설명을 참조하십시오.

데이터 유형 맵핑

연합 서버가 데이터 소스 테이블 및 뷰의 컬럼으로부터 데이터를 검색하기 위해서는, 데이터 소스에 있는 컬럼의 데이터 유형이 연합 데이터베이스에 이미 정의되어 있는 해당 데이터 유형에 맵핑되어야 합니다. DB2는 대부분의 데이터 유형 종류에 대한 기본 맵핑을 제공합니다. 예를 들어, Oracle 유형 FLOAT는 기본으로 DB2 유형 DOUBLE에 맵핑되어야 하며, OS/390용 DB2 Universal Database 유형 DATE는 기본으로 DB2 유형 DATE에 맵핑되어야 합니다. DB2 연합 서버가 지원하지 않는 데이터 유형인 LONG VARCHAR, LONG VARGRAPHIC, DATALINK, LARGE OBJECT(LOB) 유형 및 사용자 정의 유형에 대한 맵핑은 없습니다.

기본 데이터 유형 맵핑의 목록은 1413 페이지의 『기본 데이터 유형 맵핑』에서 참조하십시오.

데이터 소스 컬럼의 값이 리턴될 경우, 그 값들은 컬럼에 적용되는 유형 맵핑의 DB2 유형을 완전히 따릅니다. 이 맵핑이 기본값이면, 값들도 맵핑의 데이터 소스 유형을 완전히 따릅니다. 예를 들어, FLOAT 컬럼을 가진 Oracle 테이블이 연합 데이터베이스에 정의될 때, Oracle FLOAT의 DB2 DOUBLE에 대한 기본 맵핑은 겹쳐쓰지 않는 한 그 컬럼에 자동으로 적용됩니다. 그 결과, 컬럼으로부터 리턴된 값들은 FLOAT 및 DOUBLE을 전적으로 준수합니다.

값들이 준수해야 하는 DB2 유형을 변경함으로써 리턴된 값들의 포맷이나 길이를 변경할 수 있습니다. 예를 들어, Oracle 유형 DATE는 시간소인에 대한 것입니

다. 기본으로, 이는 DB2 유형 TIMESTAMP에 맵핑됩니다. 여러 개의 Oracle 테이블 컬럼에 DATE의 데이터 유형이 있으며, 사용자가 이들 컬럼의 조회에서 시간만 내보내기를 바란다고 가정합시다. 그런 후 사용자는 기본값을 겹쳐쓰면서 Oracle 유형 DATE를 DB2 유형 TIME에 맵핑할 수 있습니다. 그런 식으로, 컬럼이 조회될 때 시간소인의 시간 부분만 리턴될 것입니다.

CREATE TYPE MAPPING문을 사용하여 하나 이상의 데이터 소스에 적용하는 수정된 데이터 유형 맵핑을 작성할 수 있습니다. ALTER NICKNAME문을 사용하여 특정 테이블의 특정 컬럼에 대한 데이터 유형 맵핑을 수정할 수 있습니다.

922 페이지의 『CREATE TYPE MAPPING』, 511 페이지의 『ALTER NICKNAME』 및 *응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.

함수 맵핑, 함수 템플리트 및 함수 맵핑 옵션

연합 서버가 데이터 소스 함수를 인식하기 위해서는, 이 함수와 서버에 이미 존재하는 해당 DB2 함수간의 맵핑이 필요합니다. DB2는 기존 내장 데이터 소스 함수와 내장 DB2 함수간의 기본 내장을 제공합니다. 연합 서버가 인식하지 않는 데이터 소스 함수(예: 새로운 내장 함수나 사용자 정의 함수)를 사용자가 사용하길 원하면, 사용자는 이 함수와 연합 데이터베이스에서 대응되는 부분 사이의 맵핑을 작성해야 합니다. 상대편이 존재하지 않으면, 사용자가 다음 요건을 만족시키는 것을 하나 작성해야 합니다.

- 데이터 소스 함수에 입력 매개변수가 있으면, 상대편에 데이터 소스 함수가 가지고 있는 것과 동일한 갯수의 입력 매개변수가 있어야 합니다. 데이터 소스 함수에 입력 매개변수가 없으면, 상대편에도 있을 수가 없습니다.
- 입력 매개변수(있는 경우)에 대한 상대편의 데이터 유형과 리턴된 값은 데이터 소스 함수의 해당 데이터 유형과 호환되어야 합니다.

상대편은 완전한 함수 또는 함수 템플리트일 수 있습니다. *함수 템플리트*는 실행 가능 코드를 가지고 있지 않는 부분적인 함수입니다. 이는 독립적으로 호출될 수 없습니다. 유일한 용도는 데이터 소스 함수와의 맵핑에 참여하여 연합 서버로부터 데이터 소스 함수가 호출될 수 있게 하는 것입니다.

함수 맵핑은 CREATE FUNCTION MAPPING문으로 작성됩니다. 이 명령문은 사용자가 작성되고 있는 맵핑이나 맵핑 내의 데이터 소스 함수에 속하는 값을 지

정할 수 있는 *함수 맵핑 옵션*이라고 하는 매개변수를 포함합니다. 예를 들어, 그러한 값은 데이터 소스 함수가 호출될 때 소모될 오버헤드에 대한 예상 통계를 포함할 수 있습니다. 최적화 알고리즘은 함수 호출을 위한 전략 개발에 이들 측정치를 사용합니다.

함수 템플리트 및 함수 맵핑 작성에 대한 714 페이지의 『CREATE FUNCTION (소스 또는 템플리트)』 및 735 페이지의 『CREATE FUNCTION MAPPING』에서 자세한 내용을 참조하십시오. 함수 맵핑 옵션 및 그 값들에 대한 1406 페이지의 『함수 맵핑 옵션』에서 설명을 참조하십시오. 데이터 소스 함수의 호출을 최적화하는 데 대한 지침은 *응용프로그램 개발 안내서*에서 참조하십시오.

별명(Nickname) 및 컬럼 옵션

클라이언트 응용프로그램이 연합 서버로 분산 요청을 제출할 때, 서버는 적절한 데이터 소스로 요청을 분배합니다. 요청은 이들 데이터 소스를 지정할 필요가 없습니다. 그 대신, 이것은 데이터 소스에서 테이블 및 뷰 이름에 맵핑되는 별명에 의해 데이터 소스 테이블과 뷰를 참조합니다. 맵핑은 데이터 소스 이름으로 별명을 규정화해야 할 필요를 없앱니다. 테이블 및 뷰의 위치는 클라이언트 응용프로그램에 투명합니다.

별명(Nickname)은 별명(aliase)과 같은 식으로 테이블 및 뷰에 대한 대체 이름입니다. 이들은 연합 서버가 이들 오브젝트를 참조하는 포인터입니다. 별명(Nickname)은 CREATE NICKNAME문으로 정의됩니다. 764 페이지의 『CREATE NICKNAME』에서 자세한 내용을 참조하십시오.

테이블이나 뷰에 대한 별명(nickname)이 작성될 때, 최적화 알고리즘이 테이블이나 뷰에 대한 액세스를 쉽게 하기 위해 사용할 수 있는 메타데이터로 카탈로그가 채워집니다. 예를 들어, 테이블이나 뷰 컬럼의 데이터 유형이 맵핑하는 DB2 데이터 유형의 이름이 카탈로그에 제공됩니다. 별명(nickname)이 색인을 가진 테이블에 대한 것이면, 예를 들어, 색인 키에 있는 각 컬럼의 이름인 색인에 관련된 정보 역시 카탈로그에 제공됩니다.

별명(nickname)이 작성된 후, 사용자는 카탈로그에 최적화 알고리즘을 위한 더 많은 메타데이터를 제공할 수 있습니다. 예를 들어, 메타데이터는 별명이 참조하는 테이블이나 뷰의 특정 컬럼에 있는 값을 설명합니다. 사용자는 이 메타데이터를 컬

럼 옵션이라고 하는 매개변수에 지정합니다. 테이블 컬럼이 숫자 문자열만 포함하면, 사용자는 NUMERIC_STRING이라는 컬럼 옵션에 'Y' 값을 지정함으로써 이를 지시할 수 있습니다. 그 결과, 최적화 알고리즘은 데이터 소스에서 이들 문자열이 정렬되게 하여, 이들을 연합 서버에 포팅하여 거기서 정렬함으로써 생기는 오버헤드를 줄이는 전략을 형성할 수 있습니다. 값을 포함하는 데이터베이스가 연합 데이터베이스의 조합 순서와는 다른 조합 순서일 경우 그러한 절약은 특히 더 큽니다.

컬럼 옵션은 ALTER NICKNAME문으로 정의됩니다. 이 명령문에 대한 511 페이지의 『ALTER NICKNAME』에서 자세한 정보를 참조하십시오. 컬럼 옵션 및 그 설정값에 대한 설명은 1405 페이지의 『컬럼 옵션』에서 참조하십시오.

색인 스펙

데이터 소스 테이블에 대해 별명(nickname)이 작성될 때, 연합 서버는 카탈로그에 데이터 소스 테이블이 가지고 있는 모든 색인에 대한 정보를 공급합니다. 최적화 알고리즘은 이 정보를 사용하여 테이블의 데이터를 검색합니다. 테이블에 색인이 없어도, 사용자는 색인 정의가 전형적으로 포함하고 있는 정보(가령, 정보를 신속히 찾기 위해 테이블에서 어느 컬럼(들)을 검색할지)를 공급합니다. 사용자는 정보를 포함하고 테이블의 별명을 참조하는 CREATE INDEX문을 수행하여 이를 수행합니다.

사용자는 최적화 알고리즘에 연합 서버가 알지 못하는 색인을 가지고 있는 테이블에 대해 유사한 정보를 공급할 수 있습니다. 예를 들어, 색인을 가지고 있지 않지만 나중에 획득할 테이블에 대해 별명 NICK1이 작성되거나, 색인을 가지고 있는 테이블에 대한 뷰에 대해 별명 NICK2가 작성된다고 가정해 봅시다. 이런 경우, 연합 서버는 색인을 알지 못할 것입니다. 그러나 사용자는 CREATE INDEX문을 사용하여 색인이 존재한다는 것을 서버에 알려 줄 수도 있습니다. 한 명령문은 NICK1을 참조할 것이며 NICK1이 식별하는 테이블의 색인에 관한 정보를 포함할 것입니다. 다른 것은 NICK2를 참조하며 NICK2가 식별하는 뷰의 기초가 되는 기본 테이블의 색인에 대한 정보를 포함할 것입니다.

방금 설명한 것처럼, CREATE INDEX문의 정보는 색인 스펙이라고 하는 메타데이터 세트로 카탈로그화됩니다. 명령문이 별명(nickname)을 참조할 때에는 실제 색

인이 아니라 색인 스펙만을 만들어 낸다는 것을 유념하십시오. 740 페이지의 『CREATE INDEX』 및 *관리 안내서: 성능*에서 자세한 정보를 참조하십시오.

분산 요청

분산 요청은 부속 조회 및 조인 부속 선택과 같은 장치를 사용하여 어느 테이블이나 뷰 컬럼을 액세스할지와 어느 데이터를 검색할지를 지정할 수 있습니다.

이 절에서는 연합 서버가 OS/390용 DB2 Universal Database 데이터 소스, AS/400용 DB2 Universal Database 데이터 소스 및 Oracle 데이터 소스를 액세스하도록 구성되는 시나리오의 컨텍스트 내의 예를 제공합니다. 직원 정보를 포함하는 테이블이 각 데이터 소스에 보관됩니다. 연합 서버는 테이블이 상주하는 곳 즉, UDB390_EMPLOYEES, AS400_EMPLOYEES 및 ORA_EMPLOYEES를 참조하는 별명(nickname)으로 이들 테이블을 참조합니다. 사원 정보 테이블 외에도, Oracle 데이터 소스는 사원들이 있는 국가에 대한 정보를 포함하는 테이블이 있습니다. 이 두 번째 테이블에 대한 별명은 ORA_COUNTRIES입니다.

부속 조회를 가진 요청

테이블 AS400_EMPLOYEES에는 아시아에 살고 있는 직원들의 전화번호가 들어 있습니다. 여기에는 이들 전화 번호와 연관된 국가 코드도 들어 있으나, 코드가 나타내는 국가들을 나열하지는 않습니다. 그러나, 테이블 ORA_COUNTRIES는 코드와 국가 둘다를 나열합니다. 다음 조회는 부속 조회를 사용하여 중국에 대한 국가 코드를 찾아냅니다. 그리고 SELECT 및 WHERE절을 사용하여 전화 번호가 이 특정 코드를 필요로 하는 AS400_EMPLOYEES 내의 직원들을 나열합니다.

```
SELECT NAME, TELEPHONE
  FROM DJADMIN.AS400_EMPLOYEES
  WHERE COUNTRY_CODE IN
  (SELECT COUNTRY_CODE
     FROM DJADMIN.ORA_COUNTRIES
     WHERE COUNTRY_NAME = 'CHINA')
```

위에 있는 것과 같은 분산 요청이 컴파일될 때, 컴파일러의 조회 재작성 기능이 이를 보다 쉽게 최적화될 수 있는 형태로 변형합니다.

조인을 위한 요청

관계형 조인은 두 개 이상의 테이블로부터 검색된 컬럼들의 조합을 포함하는 결과 집합을 만들어 냅니다. 결과 집합의 행 *크기*를 제한하는 조건이 항상 지정되어야 합니다.

아래의 조회는 두 개의 테이블에 나열된 국가 코드를 비교하여 직원 이름과 그들의 국가 이름을 결합합니다. 각 테이블은 서로 다른 데이터 소스에 상주합니다.

```
SELECT T1.NAME, T2.COUNTRY_NAME
  FROM DJADMIN.UDB390_EMPLOYEES T1, DJADMIN.ORA_COUNTRIES T2
  WHERE T1.COUNTRY_CODE = T2.COUNTRY_CODE
```

보충

보충은 SQL문을 지원하지 않는 RDBMS에 대해 그러한 SQL문을 처리하는 것입니다. RDBMS(AS/400용 DB2 Universal Database, OS/390용 DB2 Universal Database, Oracle 등등)의 각 유형은 SQL의 국제 표준의 부속 집합를 지원합니다. 또한, 일부 유형은 이 표준을 초과하는 SQL 구조체를 지원합니다. RDBMS의 유형이 지원하는 SQL 전체를 *SQL dialect*라 부릅니다. SQL 구조체가 DB2의 SQL dialect에서 발견되나, 데이터 소스의 dialect에서는 발견되지 않으면, 연합 서버가 데이터 소스를 대신에 이 구조체를 구현할 수 있습니다.

예 1: DB2의 SQL이 절, 공통 테이블 표현식을 포함합니다. 이 절에서, 이름은 fullselect가 결과 세트를 참조할 수 있는 모든 FROM 절에서 지정할 수 있습니다. 연합 서버는 Oracle의 SQL dialect가 공통 테이블 표현식을 포함하지 않더라도, Oracle 데이터베이스에 대한 공통 테이블 표현식을 처리할 것입니다.

예 2: 응용프로그램 내에서 여러 개의 열려 있는 커서를 지원하지 않는 데이터 소스에 연결시, 연합 서버는 데이터 소스에 대해 독립적이고 동시적인 연결을 형성함으로써 이 기능을 시뮬레이트할 수 있습니다. 이와 유사하게, 연합 서버는 그 기능을 제공하지 않는 데이터 소스에 대한 CURSOR WITHHOLD 기능을 시뮬레이트할 수 있습니다.

보충은 DB2의 SQL dialect를 사용하여 연합 서버에 의해 지원되는 모든 조회를 할 수 있게 합니다. DB2외의 다른 RDBMS에 고유한 dialect를 사용할 필요는 없습니다.

통과

사용자는 통과 함수를 사용하여 데이터 소스의 고유 SQL dialect에 있는 데이터 소스와 통신할 수 있습니다. 통과에서, 사용자는 조회뿐만 아니라, DML 및 DDL 명령문도 제출할 수 있습니다. DB2 및 데이터 소스가 통과 세션에서 제출된 명령문을 처리하는 방법에 대해서는 1415 페이지의 『통과 세션에서의 SQL 처리』에서 자세한 내용을 참조하십시오.

연합 서버는 통과 세션을 관리하기 위해 다음 SQL문을 제공합니다.

SET PASSTHRU
: 통과 세션을 열고 종료합니다.

GRANT(서버 특권)
: 사용자, 그룹, 권한 부여 ID의 목록 또는 PUBLIC에 특정 데이터 소스와의 통과 세션을 시작할 수 있는 권한을 권한 부여합니다.

REVOKE(서버 특권)
: 통과 세션을 시작할 수 있는 권한을 취소합니다.

통과 사용에 특정 제한사항이 있습니다. 예를 들어, 통과 세션에서, 커서는 데이터 소스 오브젝트에 대해 직접 열 수 없습니다. 제한사항의 전체 목록은 1416 페이지의 『고려사항 및 제한사항』에서 참조하십시오.

문자 변환

*문자열*은 문자를 나타내는 일련의 바이트입니다. 자열에서, 모든 문자는 일반적인 코드화 표현으로 표시됩니다. 어떤 경우, 이러한 문자를 다른 코드화 표현으로 변환시켜야 합니다. 이런 프로세스를 *문자 변환(character conversion)*이라고 합니다. [12]

문자 변환은 SQL문을 원격으로 수행할 때 발생합니다. 예를 들어, 다음 두 경우를 생각해 보십시오.

12. 문자 변환은 필요할 때 자동으로 수행되고, 성공하면 응용프로그램에 투명해집니다. 그러므로, 명령문의 수행에 포함된 모든 문자열이 같은 방식으로 표현되면 변환 관련 정보는 필요하지 않습니다. 이것은 *독립형(standalone)* 설치와 동일한 국가내의 네트워크 같은 경우에 자주 발생합니다. 따라서, 대다수 독자의 경우 문자 변환과 관련이 없습니다.

- 응용프로그램 리퀘스터에서 응용프로그램 서버(AS)로 보내진 호스트 변수 값.
- 응용프로그램 서버(AS)에서 응용프로그램 리퀘스터로 보내진 결과 컬럼값.

이런 경우, 문자열은 전송하는 시스템과 수신하는 시스템에서 다르게 표현됩니다. 또한 같은 시스템에서의 문자열 조작시 변환이 발생할 수 있습니다.

다음 문자 변환을 설명할 때 사용되는 용어에 대해 정의합니다.

문자 세트
정의된 문자 집합. 예를 들면, 다음 문자 세트는 여러 코드 페이지에서 나타납니다.
- 26개의 액센트가 없는 A - Z
- 26개의 액센트가 없는 a - z
- 숫자 0 - 9
- . , : ; ? ( ) ' " / - _ & + % * = < >

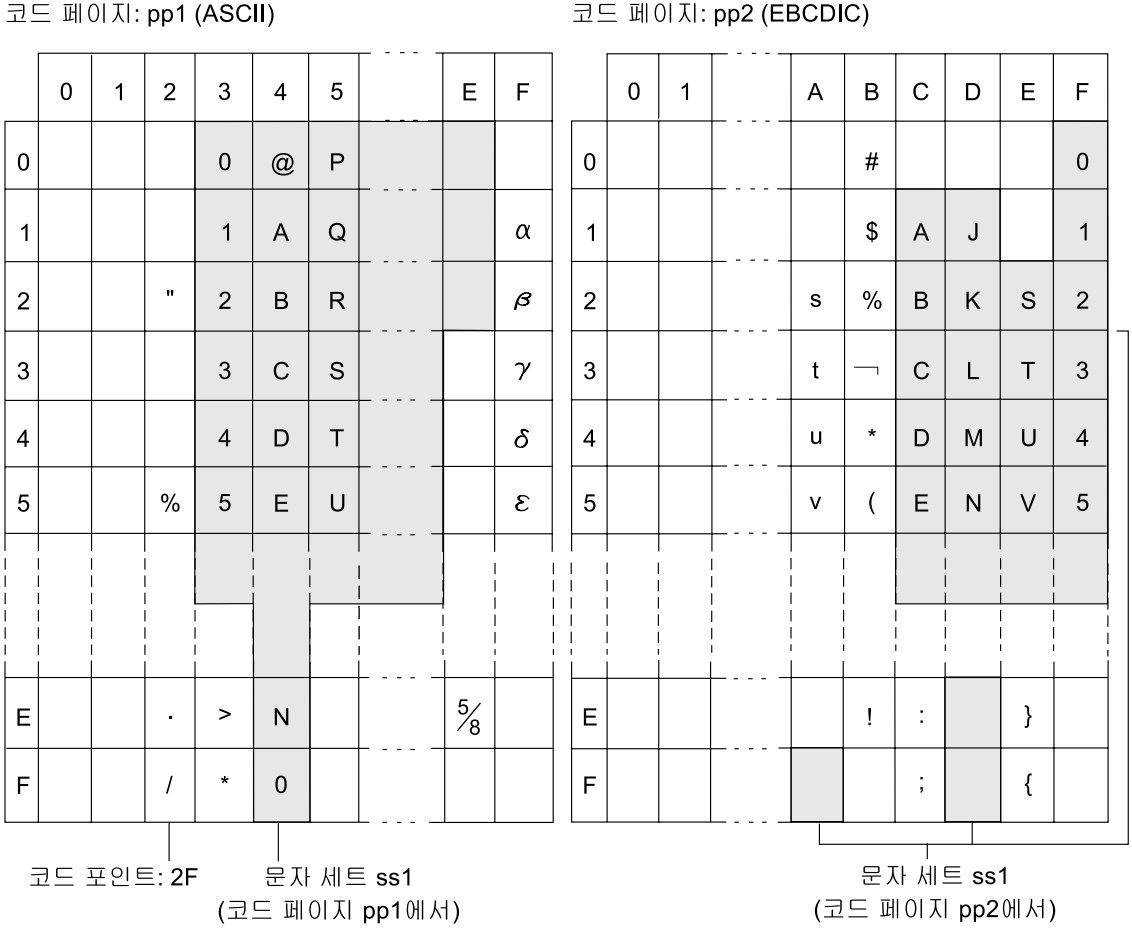
코드 페이지
코드 값이 지정된 문자 집합. 코드 페이지 850에 대한 ASCII 코드화 체계에서, 가령 "A"에는 코드 포인트 X'41'이 지정되고 "B"에는 코드 포인트 X'42'가 지정됩니다. 코드 페이지 내에서, 각 코드 값에는 하나의 특정 의미만 있습니다. 코드 페이지는 데이터베이스의 속성입니다. 응용프로그램이 데이터베이스에 연결되면, 데이터베이스 관리 프로그램은 응용프로그램의 코드 페이지를 결정합니다.

코드 포인트
문자를 나타내는 고유한 비트 패턴.

코드화 체계
문자 데이터를 표현하는 데 사용되는 규칙 집합. 예를 들면,
- 1바이트 ASCII
- 1바이트 EBCDIC
- 2바이트 ASCII
- 1바이트와 2바이트 혼합 ASCII

문자 집합과 코드 페이지

다음 예는 하나의 일반적인 문자 집합이 두 개의 다른 코드 페이지에 있는 다른 코드값으로 대응되는 방법을 보여줍니다.



동일한 코드화 체계에서 조차도, 많은 서로 다른 코드 페이지가 있으며 동일한 코드 포인트가 서로 다른 코드 페이지에서 서로 다른 문자를 표현할 수 있습니다. 게다가, 문자열의 한 바이트는 반드시 1바이트 문자 집합(SBCS)의 한 문자를 표현하지 않습니다. 문자열은 또한 혼합된 비트 데이터에 대해서도 사용됩니다. *혼합 데이터*는 1 바이트, 2 바이트 또는 다중 바이트 문자의 혼합입니다. *비트 데이터*(FOR BIT DATA 또는 BLOB로 정의된 컬럼이나, 2진 문자열)는 어떤 문자 세트와도 연관되지 않습니다.

코드 페이지 속성

데이터베이스 관리 프로그램은 응용프로그램을 데이터베이스에 바인드할 때 모든 문자열에 대한 코드 페이지 속성을 결정합니다. 잠재적인 코드 페이지 속성은 다음과 같습니다.

데이터베이스 코드 페이지 데이터베이스 구성 파일에 저장된 데이터베이스 코드 페이지. 이 코드 페이지 값은 데이터베이스를 작성할 때 결정되어 변경할 수 없습니다.

응용프로그램 코드 페이지 응용프로그램이 실행되는 코드 페이지. 이것은 응용프로그램이 바인드되는 것과 같을 필요는 없습니다(응용프로그램의 바인딩 및 실행에 대한 *응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.).

코드 페이지 0 이것은 FOR BIT DATA 또는 BLOB값이 들어 있는 표현식에서 얻은 문자열을 나타냅니다.

문자열 코드 페이지 속성

문자열 코드 페이지 속성은 다음과 같습니다.

- 컬럼은 데이터베이스 코드 페이지나 코드 페이지 0(문자 FOR BIT DATA 또는 BLOB로 정의된 경우)에 있을 수 있습니다.
- 상수 및 특수 레지스터(예, USER, CURRENT SERVER)는 데이터베이스 코드 페이지로 되어 있습니다. 상수는 SQL문이 데이터베이스에 바인드될 때 데이터베이스 코드 페이지로 변환됩니다.
- 입력 호스트 변수는 응용프로그램 코드 페이지에 있습니다.

일렬의 규칙들이 스칼라 조작의 결과, 병합 또는 세트 조작같은 문자열 오브젝트를 결합하는 동작에 대한 코드 페이지 속성을 결정하는 데 사용됩니다. 실행시, 코드 페이지 속성은 문자열의 코드 페이지 변환에 필요한 요구사항을 결정하는 데 사용됩니다.

문자 변환에 대한 세부사항은 다음을 참조하십시오.

- 문자열 지정 규칙에 대해서는 112 페이지의『문자열 지정을 위한 변환 규칙』

- 문자열의 비교나 결합시 변환 규칙에 대해서는 129 페이지의『문자열 변환 규칙』

권한 부여 및 특권

*권한 부여(Authorization)*로 사용자나 사용자 그룹이 데이터베이스 연결, 테이블 작성 또는 시스템 관리같은 일반적인 작업을 수행할 수 있습니다. *특권(privilege)*은 사용자나 그룹에 지정된 방법으로 특정 데이터베이스 오브젝트를 액세스할 수 있는 권한을 줍니다.

데이터베이스 관리 프로그램에서는 사용자가 특정 작업을 수행하는 데 필요한 각 데이터베이스 함수를 사용할수 있게 명시적으로 또는 내재적으로 권한을 부여받도록 요구합니다. [13] 따라서 테이블을 작성하려면 사용자는 테이블을 작성할 권한을 가지고 있어야 하며, 테이블을 변경하려면 사용자는 테이블을 변경할 권한을 가지고 있어야 합니다.

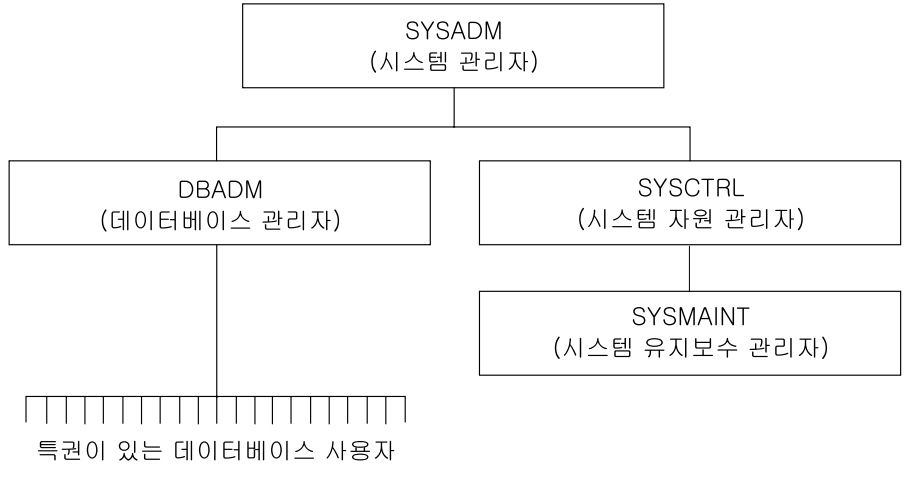


그림 7. 권한 및 특권의 계층

13. 명시적인 권한이나 특권은 사용자(U의 GRANTEETYPE)에게 허용됩니다. 내재적 권한이나 특권은 사용자가 속한 그룹에 권한 부여됩니다(GRANTEETYPE은 G).

관리자 권한이 있는 사람은 데이터베이스 관리 프로그램의 관리 작업을 수행하며, 데이터의 보안과 무결성에 대한 책임이 있습니다. 이들은 누가 데이터베이스 관리 프로그램에 액세스하며, 각 사용자가 어떤 범위까지 액세스할지를 관리합니다.

데이터베이스 관리 프로그램에서는 두 가지 관리자 권한을 제공합니다.

SYSADM

시스템 관리자 권한

DBADM

데이터베이스 관리자 권한

그리고 두 가지 시스템 제어 권한도 제공합니다.

SYSCTRL

시스템 제어 권한

SYSMAINT

시스템 유지보수 권한

SYSADM 권한은 최고 레벨의 권한을 가지며, 데이터베이스 관리 프로그램이 작성하고 유지보수하는 모든 자원을 관리합니다. SYSADM 권한에는 DBADM, SYSCTRL 및 SYSMAINT에 대한 모든 특권과 DBADM 권한을 부여하고 권한 취소할 권한이 포함됩니다.

DBADM 권한은 하나의 데이터베이스에 대한 관리자 권한입니다. 이 권한에는 오브젝트 작성, 데이터베이스 명령 실행 및 SQL문을 통한 테이블에 있는 데이터 액세스 특권이 포함됩니다. 또한, DBADM 권한에는 CONTROL 및 각각의 특권을 부여하거나 취소할 권한도 포함됩니다.

SYSCTRL 권한은 최고 레벨의 시스템 제어 권한으로, 시스템 자원에 영향을 미치는 조작에만 적용됩니다. 데이터를 직접 액세스할 수는 없습니다. 이 권한에는 데이터베이스를 작성, 갱신 또는 삭제하기 위한 특권, 인스턴스나 데이터베이스를 비활성으로 하기 위한 특권, 테이블 공간을 삭제하거나 작성하기 위한 특권이 포함됩니다

SYSMAINT 권한은 시스템 제어 권한의 두 번째 레벨입니다. SYSMAINT 권한이 있는 사용자는 인스턴스와 연관된 모든 데이터베이스에서 유지보수 조작을 수행할 수 있습니다. 데이터를 직접 액세스할 수는 없습니다. 이 권한에는 데이터베이스 구성 파일의 갱신, 데이터베이스나 테이블 공간의 백업, 기존 데이터베이스의 복원 및 데이터베이스를 모니터할 수 있는 특권이 포함됩니다.

데이터베이스 권한은 관리자가 데이터베이스 오브젝트의 특정 인스턴스에 적용되지 않는 데이터베이스 내에서, 사용자에게 수행하도록 허용한 작업에 적용됩니다. 예를 들면, 사용자가 테이블을 작성하지 않지만, 패키지를 작성할 권한이 주어진 경우입니다.

특권은 관리자나 오브젝트의 소유자가 그 데이터베이스 오브제트에 대해 수행하도록 허용한 작업에 적용됩니다. 특권이 있는 사용자는 오브젝트를 작성할 수 있습니다. 사용자는 SYSADM이나 DBADM같은 권한이 있는 사용자와는 달리 약간의 제한을 받을 수 있습니다. 예를 들어, 사용자가 테이블에 뷰를 작성하지만 트리거하지 않는 특권이 있다고 합시다. 특권이 있는 사용자는 자신의 오브젝트에 액세스하고, 자신의 오브젝트에 대한 특권을 GRANT문을 사용하는 다른 사용자에게 전달할 수 있습니다.

CONTROL 특권으로 사용자는 원하는 특정 데이터베이스 오브젝트에 액세스할 수 있고, GRANT 및 REVOKE 특권으로 그 오브젝트를 다른 사용자에게 또는 다른 사용자로부터 액세스할 수 있습니다. DBADM 권한이 CONTROL 특권을 부여하는 데 필요합니다.

개개의 특권과 데이터베이스 권한 부여로 특정 함수를 허용하지만 다른 사용자에게 같은 특권을 허용할 수 있는 권리는 포함하지 않습니다. 다른 사용자에게 테이블, 뷰 또는 스키마 특권을 부여하기 위한 권한은 GRANT문에서 WITH GRANT OPTION을 사용하여 다른 사용자에게 확대할 수 있습니다.

테이블 공간 및 기타 저장영역 구조

저장영역 구조(storage structures)에는 데이터베이스의 오브젝트가 들어 있습니다. 데이터베이스 관리 프로그램이 관리하는 기본 저장영역 구조는 테이블 공간입니다. *테이블 공간*은 테이블, 색인, 대형 오브젝트(LOB) 및 데이터 유형이 LONG인 데이터가 들어 있는 저장영역 구조입니다. 두 가지 유형의 테이블 공간이 있습니다.

데이터베이스 관리 공간(DMS) 테이블 공간
: 데이터베이스 관리 프로그램에 의해 관리되는 공간이 있는 테이블 공간.

시스템 관리 공간(SMS) 테이블 공간
: 운영 시스템에 의해 관리되는 공간이 있는 테이블 공간.

모든 테이블 공간은 컨테이너로 구성됩니다. *컨테이너*는 테이블같은 오브젝트가 저장되는 위치를 설명합니다. 예를 들어, 파일 시스템에 있는 서브디렉토리가 컨테이너가 될 수 있습니다.

테이블 공간과 컨테이너에 대한 861 페이지의『CREATE TABLESPACE』또는 *관리 안내서*에서 자제한 정보를 참조하십시오.

테이블 공간 컨테이너로부터 읽어온 데이터는 *버퍼 풀(buffer pool)*이라고 하는 메모리 영역에 저장됩니다. 버퍼 풀은 테이블 공간과 연관되어 있어, 데이터를 버퍼에 저장할 때 동일한 메모리 영역을 공유할 데이터를 제어할 수 있게 합니다. 버퍼 풀에 대한 631 페이지의『CREATE BUFFERPOOL』또는 *관리 안내서*에서 자세한 정보를 참조하십시오.

파티션된 데이터베이스는 데이터가 다른 데이터베이스 파티션에 분포되도록 해줍니다. 이러한 파티션에 포함된 파티션은 테이블 공간에 지정된 노드 그룹에 의해 결정됩니다. 노드 그룹은 데이터베이스의 일부로 정의된 하나 이상의 파티션으로 구성된 그룹입니다. 테이블 공간에는 노드 그룹의 각 파티션에 대한 컨테이너가 하나 이상 포함되어 있습니다. *파티션 맵*은 각 노드 그룹과 연관되어 있습니다. 파티션 맵은 주어진 데이터 행을 저장하게 될 노드 그룹의 파티션을 결정하기 위해 데이터베이스 관리 프로그램이 사용합니다. 노드 그룹과 데이터 파티션에 대한 68 페이지의『여러 파티션에 걸친 데이터 파티션』, 768 페이지의『CREATE NODEGROUP』또는 *관리 안내서*에서 자세한 정보를 참조하십시오.

테이블에는 외부 파일에 저장된 데이터로의 링크를 등록하는 컬럼이 포함될 수도 있습니다. DATALINK 데이터 유형입니다. 일반 테이블에 기록된 DATALINK 값은 외부 파일 서버에 저장된 파일을 가리킵니다.

파일 서버에 설치된 DB2 Data Links Manager 제품은 DB2와 결합하여 다음과 같은 선택적 기능을 제공합니다.

- 현재 DB2에 연결된 파일이 삭제되거나 재명명되지 않도록 하기 위한 참조 무결성.
- DATALINK 컬럼에서 적절한 SQL 특권을 가진 컬럼만이 그 컬럼에 연결된 파일을 읽을 수 있도록 하는 보안.
- 파일의 조정 백업 및 복구.

DB2 Data Links Manager 제품에는 다음과 같은 기능이 들어 있습니다.

데이터 링크 파일 관리 프로그램
DB2에 연결된 특정 파일 서버에 있는 모든 파일을 등록합니다.

데이터 링크 파일 시스템 필터

등록된 파일이 삭제되거나 재명명되지 않도록 파일 시스템 명령을 거릅니다. 또한, 적절한 액세스 권한이 존재하도록 명령을 거르기도 합니다.

DB2 Data Links Manager에 대한 *관리 안내서*에서 자세한 정보를 참조하십시오.

여러 파티션에 걸친 데이터 파티션

파티션된 데이터베이스의 여러 파티션(노드)에 걸쳐 데이터를 분산시키는 데 있어서 DB2에는 상당한 융통성이 있습니다. 사용자는 파티션 키를 선정하여 데이터 파티션 방식을 선택할 수 있으며, 노드 그룹과 데이터가 저장될 테이블 공간을 선택하여 테이블 데이터가 분산될 파티션과 개수를 결정할 수 있습니다. 또한,(사용자가 갱신할 수 있는) 파티션 맵은 파티션에 대한 파티션 키 값의 맵핑을 지정합니다. 그러면, 대형 테이블을 위한 파티션된 데이터베이스 전반에 걸쳐 워크로드를 융통성있게 배분할 수 있으며, 응용프로그램 개발자가 선택할 경우 소형 테이블을 하나 또는 소수의 파티션에 저장할 수 있습니다. 각 지역 파티션에는 고성능의 지역 데이터 액세스를 제공하기 위해, 저장되는 데이터에 대한 지역 색인이 표시됩니다.

파티션된 데이터베이스는 파티션된 저장영역 모델을 지원하는 데, 여기서는 일련의 데이터베이스 파티션에 걸쳐 테이블 데이터를 파티션할 때 파티션 키가 사용됩니다. 색인 데이터도 해당 테이블과 함께 파티션되며, 각 파티션에 지역적으로 저장됩니다.

데이터베이스 데이터를 저장하는 데 파티션을 사용하려면, 이들 파티션들이 데이터베이스 관리 프로그램에 정의되어 있어야 합니다. 파티션은 파일 `db2nodes.cfg`에서 정의됩니다. 파티션 정의에 대한 *관리 안내서*에서 자세한 내용을 참조하십시오.

파티션된 노드 그룹상의 테이블 공간에 있는 테이블의 파티션 키는 CREATE TABLE문(또는 ALTER TABLE문)에서 지정됩니다. 파티션 키가 지정되지 않을 경우, 기본적으로 기본 키의 첫번째 컬럼부터 테이블에 대한 파티션 키가 작성됩니다. 기본 키가 지정되지 않을 경우, 데이터 유형이 LONG이나 LOB가 아닌 테이블에 정의된 첫번째 컬럼이 기본 파티션 키가 됩니다. 파티션 테이블에는 데이터

유형이 LONG이나 LOB가 아닌 컬럼이 적어도 하나는 있어야 합니다. 단일 파티션 노드 그룹의 테이블 공간에 있는 테이블은 명시적으로 지정되는 경우에만 파티션 키를 갖습니다.

*해쉬 파티션(hash partitioning)*은 다음과 같이 파티션상에 행을 배치하는 경우 사용됩니다.

1. 해쉬 알고리즘(파티션 함수)은 파티션 맵 색인이 생성되는 파티션 키의 모든 컬럼에 적용됩니다.
2. 이 파티션 맵 색인은 파티션 맵에 대한 색인으로 사용됩니다. 파티션 맵의 해당 색인에 있는 파티션 번호가 행이 저장되는 파티션입니다.
3. 파티션 맵은 노드 그룹과 관련되며, 테이블은 노드 그룹상의 테이블 공간에 작성됩니다.

DB2는 *부분 클러스터 해제(partial declustering)*를 지원하는 데, 이는 시스템 내의 부속 집합 파티션에 걸쳐 테이블을 파티션할 수 있음을 의미합니다. 시스템의 모든 파티션에 걸쳐 테이블을 파티션할 필요가 없습니다.

파티션 맵

각 노드 그룹은 *파티션 맵*과 연관되어 있는데, 이는 4 096개의 파티션 번호 배열입니다. 테이블의 각 행에 대한 파티션 함수에 의해 생성되는 파티션 맵 색인은 행이 저장되는 파티션을 결정하는 데 사용됩니다.

70 페이지의 그림8은 파티션 키 값(c1, c2, c3)을 갖는 행이 파티션 맵 색인 2(파티션 p5 참조)로 맵핑되는 방식을 보여줍니다.

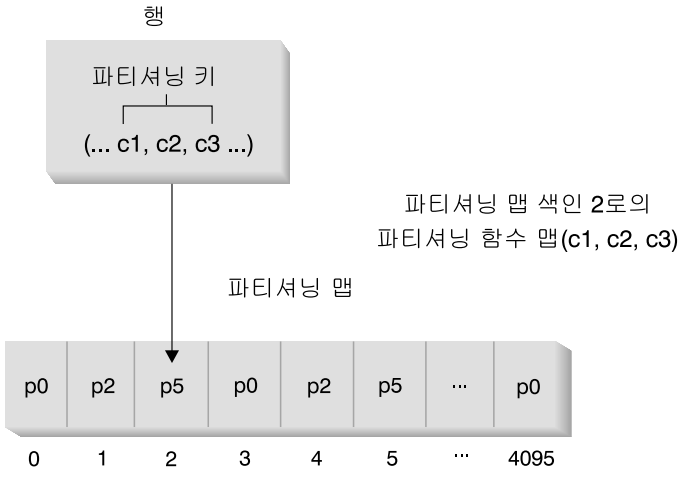


그림 8. 데이터 분산

파티션 맵을 변경할 수 있으므로, 파티션 키나 실질적인 데이터를 변경하지 않고도 데이터 분산을 변경할 수 있습니다. REDISTRIBUTE NODEGROUP 명령의 일부 또는 새로운 파티션 맵을 노드 그룹의 테이블 재분산시 사용하는 API로 새로운 파티션 맵이 지정됩니다. *Command Reference* 또는 *Administrative API Reference*에서 자세한 정보를 참조하십시오.

테이블 배열

DB2에는 조인이나 부속 조회를 위해 액세스한 데이터가 동일한 노드 그룹의 동일 파티션에 놓이게 되는 시점을 인식하는 능력이 있습니다. 위와 같은 경우, DB2는 데이터가 저장된 파티션에서 조인이나 부속 조회 처리 중 하나를 수행하도록 선택할 수 있으며, 이는 상당한 성능상의 이점이 있습니다. 이러한 경우를 테이블 배열이라 합니다. 배치된 테이블은 다음과 같아야 합니다.

- 동일한 노드 그룹에 있어야 한다(재분배 되지 않은).[14]
- 컬럼 수가 동일한 파티션 키가 있어야 합니다.

14. 노드 그룹에 있는 노드 그룹, 테이블의 재분배로 인해 다른 파티션 맵을 사용하게 되면, 재분배되지 않는다. 즉, 테이블은 배치되지 않는다.

- 파티션 키의 해당 컬럼이 파티션과 호환 가능해야 합니다(132 페이지의 『파티션 호환성』 참조).

또는

- 동일한 파티션에 정의된 단일 파티션 노드 그룹에 존재해야 합니다.

동일한 파티션 키 값을 갖는 배열된 테이블의 행은 동일한 파티션에 배치됩니다.

제3장 언어 요소

이 장에서는 SQL문의 기본적인 구문과 많은 SQL문의 공통적인 언어 요소를 정의합니다.

문자

SQL 언어에 있는 기본적인 키워드와 연산자 기호는 모든 IBM 문자 집합의 일부인 1바이트 문자입니다. 언어 문자(letter)는 문자, 디지트 또는 특수 문자로 구분됩니다.

*문자*는 26개의 대문자(A - Z)와 26개의 소문자(a - z), 그리고 호스트 데이터베이스 제품과의 호환성을 위해 포함되는 세 개의 문자($, #, @)중 하나입니다. 예를 들어, 코드 페이지 850에서 $는 X'24'에, #는 X'23', @는 X'40'에 있습니다. 또한 문자는 확장 문자 집합의 영문자를 포함합니다. 확장 문자 세트에는 추가 영문자가 포함됩니다. 예를 들어, 발음 구별 부호가 있는 확장 문자 세트가 있습니다. ´는 발음 구별 부호의 예입니다. 사용할 수 있는 문자는 사용 중인 코드 페이지에 따라 다릅니다.

*디지트(digit)*는 0-9까지의 문자입니다.

*특수 문자*는 다음과 같습니다.

	공백	-	마이너스 부호
"	큰따옴표	.	마침표
%	퍼센트	/	슬래쉬
&	엠퍼센드	:	콜론
'	작은 따옴표	;	세미콜론
(	왼쪽 괄호	<	보다 작다
)	오른쪽 괄호	=	같다
*	별표	>	보다 크다
+	플러스 부호	?	물음표
,	쉼표	_	밑줄
|	수직막대	^	케릿
!	느낌표		

MBCS에 관한 고려사항

모든 복수 바이트 문자(character)는 특수 문자인 2바이트 공백을 제외하고는 문자(letter)로 처리됩니다.

토큰

언어의 기본 구문 구성 단위를 토큰이라고 합니다. 토큰은 하나 이상의 일련의 문자열입니다. 토큰이 공백을 포함할 수 있는 문자 또는 분리 식별자인 경우를 제외하고는 토큰에는 공백 문자가 포함될 수 없습니다(이러한 용어는 이후에 정의됩니다.).

토큰은 *일반(ordinary)*이나 *분리 문자(delimiter)* 토큰으로 분류됩니다.

- *일반 토큰*은 숫자 상수, 일반 식별자, 호스트 식별자 또는 키워드입니다.

  예

  ```
  1       .1       +2       SELECT       E       3
  ```

- *분리 문자* 토큰은 문자, 분리 식별자, 연산자 부호 또는 구문 도표에 있는 특수 문자입니다. 물음표는 1087 페이지의 『PREPARE』에서 설명한 것처럼, 매개변수 표시문자로 제공되는 경우 또한 분리 문자 토큰입니다.

  예

  ```
  ,       'string'       "fld1"       =       .
  ```

공백: 공백은 하나 이상의 일련의 공백 문자열입니다. 문자과 분리 식별자 이외의 토큰에는 공백이 있을 수 없습니다. 어떤 토큰 다음에는 공백이 올수도 있습니다. 모든 일반 토큰은 구문에서 허용하는 경우, 뒤에 공백이나 분리 문자 토큰이 옵니다.

주석: 정적 SQL문에는 호스트 언어 주석이나 SQL 주석을 포함할 수 있습니다. 분리 문자 토큰이나 키워드 EXEC와 SQL 사이를 제외하고, 공백이 지정될 수 있는 곳이면 어디든지 이들 두 가지 유형의 주석을 지정할 수 있습니다. SQL 주석은 두 개의 연속적인 하이픈(--)으로 시작하여 행이 끝날 때 종료합니다. 507 페이지의 『SQL 주석』에서 자세한 내용을 참조하십시오.

대문자 및 소문자: 토큰에는 소문자가 포함될 수 있지만, 일반 토큰의 소문자는 대소문자 구별 식별자가 있는 C 언어의 호스트 변수의 경우를 제외하고 대문자로 변환됩니다. 분리 문자 토큰은 대문자로 변환되지 않습니다. 그러므로, 다음과 같은 명령문은

```
select * from EMPLOYEE where lastname = 'Smith';
```

변환되어 다음과 같이 됩니다.

```
SELECT * FROM EMPLOYEE WHERE LASTNAME = 'Smith';
```

MBCS에 관한 고려사항

복수 바이트 영문자는 대문자로 변환되지 않습니다. 1바이트 문자 집합(a-z)은 대문자로 변환됩니다.

식별자

*식별자*는 이름을 구성하는 데 사용되는 토큰입니다. SQL문에 있는 식별자는 SQL 식별자나 호스트 식별자입니다.

SQL 식별자

두 가지 유형의 SQL 식별자, 즉 *일반 식별자*와 *구분 식별자*가 있습니다.

- *일반 식별자*는 문자로서, 뒤에 0 또는 그 이상의 문자가 오며, 각각의 대문자, 디지트 또는 밑줄이 올 수 있습니다. 일반 식별자는 예약어와 동일할 수 없습니다(예약어에 대해서는 1443 페이지의 『부록H. 예약된 스키마 이름 및 예약어』에서 정보를 참조하십시오.).
- *구분 식별자*는 큰 따옴표로 묶인 하나 이상의 문자입니다. 2개의 연속적인 큰 따옴표는 구분 식별자 내에서 하나의 따옴표를 나타내는 데 사용됩니다. 이 방식으로 식별자에 소문자가 포함될 수 있습니다.

일반 식별자 및 분리 식별자에 대한 예는 다음과 같습니다.

```
WKLYSAL    WKLY_SAL    "WKLY_SAL"    "WKLY SAL"    "UNION"    "wkly_sal"
```

2바이트 코드 페이지에서 작성되었지만 복수 바이트 코드 페이지의 응용프로그램 또는 데이터베이스에서 사용되는 식별자 사이의 문자 변환은 특수 고려사항을 필요로 할 수도 있습니다. 복수 바이트로의 변환 후에 그러한 식별자가 식별자의 길이 한계를 초과할 수도 있습니다(자세한 설명은 1505 페이지의 『부록O. 일본어 및 대만어의 EUC 고려사항』 참조).

호스트 식별자

*호스트 식별자*는 호스트 프로그램에서 선언된 이름입니다. 호스트 식별자를 구성하는 규칙은 호스트 언어의 규칙입니다. 호스트 ID는 255자보다 길 수 없으며 'SQL' 또는 'DB2' 대문자나 소문자 철자로 시작할 수 없습니다.

명명 규칙 및 내재된 오브젝트 이름 규정

이름 구성 규칙은 이름 오브젝트 유형에 따라 다릅니다. 데이터베이스 오브젝트 이름은 하나의 식별자로 구성되거나 또는 두 개의 식별자로 구성된 스키마 규정 오브젝트일 수 있습니다. 스키마 규정 오브젝트 이름은 스키마 이름 없이 지정될 수도 있습니다. 이런 경우, 스키마 이름은 내재된 이름입니다.

동적 SQL문에서, 스키마 규정 오브젝트 이름은 내재적으로 CURRENT SCHEMA라는 특수 레지스터 값을 규정화되지 않은 오브젝트 이름 참조를 위한 규정자로 사용합니다. 기본값으로, 이는 현재 권한 부여 ID로 설정되어 있습니다. 1177 페이지의 『SET SCHEMA』에서 자세한 내용을 참조하십시오. 동적 SQL문이 DYNAMICRULES BIND 옵션으로 바인드된 패키지로부터의 명령문이면, CURRENT SCHEMA 특수 레지스터는 규정에서 사용되지 않습니다. DYNAMICRULES BIND 패키지에서, 패키지 기본 규정자는 동적 SQL문 내에서 규정되지 않은 오브젝트 참조 규정에 대한 값으로 사용됩니다.

정적 SQL에서, QUALIFIER precompiler/bind 옵션은 내재적으로 지정되지 않은 데이터베이스 오브젝트 이름에 대한 규정자를 지정합니다. 기본적으로, 이것은 패키지 권한 부여 ID로 설정됩니다. *Command Reference*에서 자세한 내용을 참조하십시오.

구문 도표는 다른 유형의 이름에 대해 다른 용어를 사용합니다. 다음 목록은 이러한 용어를 정의합니다. 여러 가지 식별자의 최대 길이는 1253 페이지의 『부록A. SQL 한계』에서 참조하십시오.

별명 별명을 지정하는 스키마 규정 이름.

속성 이름 구조화 데이터 유형의 속성을 나타내는 식별자.

권한 부여 이름 사용자나 그룹을 나타내는 식별자. 문자에 다음의 제한사항이 사용될 수 있음을 기억하십시오.

- 유효한 문자는 A - Z, a - z, 0 - 9, #, @, $, _입니다.
- 이름은 문자 'sYS', 'IBM' 또는 'sQL'로 시작할 수 없습니다.
- ADMINS, GUESTS, LOCAL, PUBLIC 또는 USERS 같은 이름은 사용할 수 없습니다.
- 분리된 권한 부여 ID에는 소문자가 포함될 수 없습니다.

버퍼풀 이름 버퍼풀을 나타내는 식별자.

컬럼 이름 테이블이나 뷰의 컬럼을 나타내는 규정화되거나 규정화되지 않은 이름. 규정자는 테이블 이름, 뷰 이름, 별명 또는 상관 이름일 수 있습니다.

조건 이름 SQL 프로시듀어의 조건을 지정하는 식별자.

제한조건 이름 참조 제한조건, 기본 키 제한조건, 고유한 제한조건 및 테이블 점검 제한조건을 나타내는 식별자.

상관 이름 결과 테이블을 나타내는 식별자.

커서 이름 SQL 커서를 나타내는 식별자. 호스트 호환성을 위해 이름에 하이픈 문자를 사용할 수도 있습니다.

데이터 소스 이름 데이터 소스를 나타내는 식별자. 이 식별자는 원격 오브젝트 이름의 세 부분 중 첫번째입니다.

설명자 이름 콜론 다음에 SQL 설명 영역(SQLDA)을 나타내는 호스트 식별자가 뒤에 옵니다. 호스트 식별자의 설명에 대해서는 156 페이지의 『호스트 변수 참조』에서 참조하십시오. 설명자 이름에는 표시기 변수가 포함될 수 없음을 기억하십시오.

구별 유형 이름 구별 유형 이름을 지정하는 규정화된 또는 규정화되지 않은 이름. SQL문 내의 규정화되지 않은 구

	별 유형 이름은 데이터베이스 관리 프로그램에 의해 문맥에 따라 내재적으로 규정화됩니다.
이벤트 모니터 이름	이벤트 모니터를 나타내는 식별자.
함수 맵핑 이름	함수 맵핑을 나타내는 식별자.
함수 이름	함수를 나타내는 규정화된 또는 규정화되지 않은 이름. SQL문에서 규정화되지 않은 함수 이름은 문맥에 따라 데이터베이스 관리 프로그램이 내재적으로 규정화합니다.
그룹 이름	구조화된 유형에 대해 정의된 변환 그룹을 지시하는 규정되지 않은 식별자.
호스트 변수	호스트 변수를 나타내는 일련의 토큰. 호스트 변수에는 156 페이지의 『호스트 변수 참조』에서 설명한 바와 같이 적어도 하나의 호스트 식별자를 포함합니다.
색인 이름	색인 또는 색인 스펙을 나타내는 스키마 규정 이름.
레이블	SQL 프로시듀어의 레이블을 지정하는 식별자.
메소드 이름	메소드를 지정하는 식별자. 메소드에 대한 스키마 문맥은 메소드의 주제 유형(또는, 주제 유형의 부속 유형)의 스키마에 의해 판별됩니다.
별명(nickname)	연합된 서버 참조를 테이블이나 뷰로 지정하는 스키마 규정 이름.
노드 그룹 이름	노드 그룹을 나타내는 식별자.
패키지 이름	패키지를 지정하는 스키마 규정 이름.
매개변수 이름	프로시듀어, 사용자 정의 함수(UDF), 메소드 또는 색인 확장에서 참조될 수 있는 매개변수를 명명하는 식별자.
프로시듀어 이름	프로시듀어를 지정하는 규정화된 또는 규정화되지 않은 이름. SQL문에 규정화되지 않은 프로시듀어

이름은 문맥에 따라 데이터베이스 관리 프로그램이 내재적으로 규정화합니다.

원격 권한 부여 이름 데이터 소스 사용자를 나타내는 식별자. 권한 부여 이름에 대한 규칙은 데이터 소스에 따라 다릅니다.

원격 함수 이름 등록된 함수를 데이터 소스 데이터베이스로 지정하는 이름.

원격 오브젝트 이름 데이터 소스 테이블이나 뷰를 지정하고 테이블이나 뷰가 상주하는 데이터 소스를 식별하는 세 부분으로 된 이름. 이 이름의 부분들은 데이터 소스 이름, 원격 테이블 이름 및 원격 테이블 이름입니다.

원격 스키마 이름 데이터 소스 테이블이나 뷰가 속하는 스키마를 지정하는 이름. 이 이름은 원격 오브젝트 이름의 세 부분 중 두 번째입니다.

원격 테이블 이름 데이터 소스에서 테이블이나 뷰를 지정하는 이름. 이 이름은 원격 오브젝트 이름의 세 부분 중 세 번째입니다.

원격 유형 이름 데이터 소스 데이터베이스에 의해 지원되는 데이터 유형. 시스템 내장 유형에는 긴 형태를 사용하지 마십시오(예를 들어, CHARACTER 대신 CHAR를 사용하십시오.).

저장 지점 이름 저장 지점을 지시하는 식별자.

스키마 이름 SQL 오브젝트에 대한 논리 그룹핑을 제공하는 식별자. 오브젝트의 규정화 이름으로 사용되는 스키마 이름은 다음으로부터 내재적으로 결정될 수 있습니다.

- CURRENT SCHEMA 특수 레지스터의 값으로부터
- QUALIFIER precompile/bind 옵션으로부터

- CURRENT PATH 특수 레지스터를 사용하는 분석 알고리즘에 따라
- 동일한 SQL문에 있는 다른 오브젝트의 스키마 이름에 따라

복잡하지 않도록 하려면, 스키마 이름 "SESSION"을 사용해야 하는 선언된 전역 임시 테이블에 대해 스키마로 사용되는 것을 제외하고는, 스키마 이름 "SESSION"을 사용하지 않는 것이 좋습니다.

서버 이름 응용프로그램 서버를 나타내는 식별자. 연합된 시스템에서는, 서버 이름이 또한 데이터 소스의 지역 이름을 지정합니다.

특정 이름 고유 이름을 지정하는 규정화된 또는 규정화되지 않은 이름. SQL문에 규정화되지 않은 특정 이름은 문맥에 따라 데이터베이스 관리 프로그램이 내재적으로 규정화합니다.

SQL 변수 이름 SQL 프로시듀어문에서 지역 변수 이름을 정의합니다. SQL 변수 이름은 호스트 변수 이름이 허용되는 다른 SQL문에서 사용할 수 있습니다. 이름은 SQL 변수를 선언한 복합 명령문의 레이블에 의해 규정될 수 있습니다.

명령문 이름 준비된 SQL문을 나타내는 식별자.

상위 유형 이름 유형 이름을 서브유형을 지정하는 규정화된 또는 규정화되지 않은 이름. SQL문에 규정화되지 않은 상위 유형 이름은 문맥에 따라 데이터베이스 관리 프로그램이 내재적으로 규정화합니다.

테이블 이름 테이블을 지정하는 스키마 규정 이름.

테이블 공간 이름 테이블 공간을 나타내는 식별자.

트리거 이름 트리거를 지정하는 스키마 규정 이름.

유형 맵핑 이름 데이터 유형 맵핑을 나타내는 식별자.

유형 이름	유형 이름을 지정하는 규정화된 또는 규정화되지 않은 이름. SQL문에 규정화되지 않은 유형 이름은 문맥에 따라 데이터베이스 관리 프로그램이 내재적으로 규정화합니다.
분류된 테이블 이름	분류된 테이블을 나타내는 스키마 규정 이름.
입력된 뷰 이름	입력된 뷰를 지정하는 스키마 규정 이름.
뷰 이름	뷰를 지정하는 스키마 규정 이름.
랩퍼 이름	랩퍼를 지정하는 식별자.

별명

테이블 별명은 테이블이나 뷰에 대한 대체 이름으로 간주될 수 있습니다. 따라서 테이블이나 뷰는 이름이나 테이블 별명으로 SQL문에서 참조할 수 있습니다.

별명은 테이블이나 뷰 이름이 사용될 수 있는 모든 곳에서 사용될 수 있습니다. 별명은 오브젝트가 없더라도(이것을 참조하는 명령문이 컴파일될 때 있어야 하더라도) 작성될 수 있습니다. 별명 체인을 따라 순환적으로 또는 반복적으로 참조하지 않는 경우, 다른 별명을 참조할 수 있습니다. 별명은 같은 데이터베이스에서 테이블, 뷰 또는 별명을 참조만 할 수 있습니다. 별명은 CREATE TABLE이나 CREATE VIEW문에서와 같이 새로운 테이블이나 뷰 이름이 예상되는 위치에서는 사용할 수 없습니다. 예를 들어, PERSONNEL 별명이 작성되면, CREATE TABLE PERSONNEL...과 같은 후속 명령문에서는 오류가 발생합니다.

별명으로 테이블이나 뷰를 참조하는 옵션은 구문 도표에서 명확하게 표시되지 않거나 SQL문의 설명에서 언급되지 않습니다.

규정화되지 않은 새로운 별명은 기존의 테이블, 뷰 또는 별명과 동일한 완전히 규정화된 이름을 가질 수 없습니다.

SQL문에서 별명을 사용하는 효과는 원문 대체 효과와 유사합니다. SQL문을 컴파일할 때 정의해야 하는 별명은 규정화된 기본 테이블이나 뷰 이름으로 명령문 컴파일시 대체됩니다. 예를 들면, `PBIRD.SALES`가 `DSPN014.DIST4_SALES_148`의 별명인 경우 컴파일할 때,

```
SELECT * FROM PBIRD.SALES
```

는 다음과 같이 됩니다.

```
SELECT * FROM DSPN014.DIST4_SALES_148
```

연합된 시스템에서는, 앞서 기술한 사용 및 제한사항이 테이블 별명(alias)뿐만 아니라 별명(Nickname)에 대한 별명(aliase)에도 적용됩니다. 그러므로, 별명(nickname)의 별명(alias)이 SQL문에서 별명(nickname)을 대신해 사용될 수 있습니다. 별명(alias)을 참조하는 명령문이 컴파일되기 전에 별명(nickname)이 작성된다면 아직 존재하지 않는 별명(nickname)에 대한 별명(alias)을 작성할 수 있습니다. 별명(nickname)에 대한 별명(alias)은 그 별명(nickname)에 대한 또 다른 별명(alias)을 참조할 수 있습니다.

다른 관계형 데이터베이스 관리 시스템 응용프로그램의 구문 허용 한도의 경우, CREATE ALIAS와 DROP ALIAS문에서 ALIAS 대신 SYNONYM을 사용할 수 있습니다.

별명에 대한 627 페이지의 『CREATE ALIAS』에서 자세한 정보를 참조하십시오.

권한 부여 ID 및 권한 부여 이름

*권한 부여 ID*는 데이터베이스 관리 프로그램과 응용프로그램 프로세스 또는 프로그램 준비 프로세스 사이에서 연결을 구축할 때 데이터베이스 관리 프로그램에 의해 얻은 문자열입니다. 이것은 특권 집합을 나타냅니다. 또한 사용자나 사용자 그룹을 나타내지만, 이 권한이 데이터베이스 관리 프로그램에 의해 제어되지는 않습니다.

권한 부여 ID는 데이터베이스 관리 프로그램에 의해 다음을 수행하는 데 사용됩니다.

- SQL문의 권한 부여 점검
- QUALIFIER precompile/bind 옵션 및 CURRENT SCHEMA 특수 레지스터의 기본값. 또한, 권한 부여 ID는 기본 CURRENT PATH 특수 레지스터 및 FUNCPATH precompile/bind 옵션에 포함됩니다.

권한 부여 ID는 모든 SQL문에 적용됩니다. 정적 SQL문에 적용되는 권한 부여 ID는 프로그램 바인딩동안 사용되는 권한 부여 ID입니다. 동적 SQL문에 적용되는 권한 부여 ID는 동적 SQL문을 발행하는 패키지에 대해 바인딩 시간에 제공되는 DYNAMICRULES 옵션을 근거로 합니다. DYNAMICRULES RUN으로 바인드된 패키지의 경우, 사용된 권한 부여 ID는 패키지를 실행하는 사용자의 권한 부여 ID입니다. DYNAMICRULES BIND로 바인드된 패키지의 경우, 권한 부여 ID는 패키지의 권한 부여 ID입니다. 이것은 *런타임 권한 부여 ID*라고 합니다.

SQL문에 지정된 *권한 부여 이름*은 명령문의 권한 부여 ID와 혼돈해서는 안 됩니다. 권한 부여 이름은 여러 가지 SQL문에서 사용되는 식별자입니다. 권한 부여 이름은 스키마의 소유자를 지정하기 위해 CREATE SCHEMA문에서 사용됩니다. 권한 부여 이름은 GRANT와 REVOKE문에서 사용되어 권한 부여나 권한 취소의 대상을 지정합니다. *X*에 대한 특권 부여는 *X* 또는 그룹 X의 구성원이 뒤에 그러한 특권을 필요로 하는 명령문의 권한 부여 ID가 될 것이라는 것을 전제로 한다는 것을 유념하십시오.

예:

- SMITH가 응용프로그램 프로세스와의 연결이 구축될 때 데이터베이스 관리 프로그램이 확보한 사용자 ID 및 권한 부여 ID라고 합시다. 다음 명령문은 대화식으로 수행됩니다.

  ```
  GRANT SELECT ON TDEPT TO KEENE
  ```

  SMITH는 명령문의 권한 부여 ID입니다. 그러므로, 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터의 기본값과 정적 SQL에서 기본 QUALIFIER 사전 컴파일/바인드 옵션은 SMITH입니다. 이에 따라, 그 명령문을 실행하기 위한 권한은 SMITH에 대해 점검되고 SMITH는 77 페이지의 『명명 규칙 및 내재된 오브젝트 이름 규정』에 설명된 규정 규칙을 기초로 하는 *테이블 이름* 내재 규정자입니다.

  KEENE은 명령문에 지정된 권한 부여 이름입니다. KEENE에게 SMITH.TDEPT에 대한 SELECT 특권이 주어집니다.
- SMITH가 관리 권한을 가지고 있고 이 SMITH가 세션 동안 SET SCHEMA 문이 발행되지 않은 다음과 같은 동적 SQL문의 권한 부여 ID라고 가정해봅시다.

```
DROP TABLE TDEPT
```

위 명령문은 SMITH.TDEPT 테이블을 제거합니다.

```
DROP TABLE SMITH.TDEPT
```

위 명령문은 SMITH.TDEPT 테이블을 제거합니다.

```
DROP TABLE KEENE.TDEPT
```

위 명령문은 KEENE.TDEPT 테이블을 제거합니다. KEENE.TDEPT와 SMITH.TDEPT가 다른 테이블임을 기억하십시오.

```
CREATE SCHEMA PAYROLL AUTHORIZATION KEENE
```

KEENE는 RAYROLL을 호출한 스키마를 작성한 명령문에 지정된 권한 부여 이름입니다. KEENE는 스키마 PAYROLL의 소유자이며 CREATEIN, ALTERIN 과 DROPIN 권한을 갖고 다른 사용자에게 그들 권한을 부여할 수 있습니다.

런타임시 동적 SQL 특성

BIND 및 PRECOMPILE 명령 옵션 OWNER는 패키지의 권한 부여 ID를 정의합니다.

BIND 및 PRECOMPILE 명령 옵션 QUALIFIER는 패키지에 포함된 비규정화된 오브젝트에 대한 내재적 규정자를 정의합니다.

BIND 및 PRECOMPILE 명령 옵션 DYNAMICRULES는 런타임 동적 SQL문이 런타임 규칙 DYNAMICRULES RUN으로 처리되는지 또는 바인드시 규칙 DYNAMICRULES BIND로 처리되는지 여부를 결정합니다. 이들 규칙은 권한 부여 ID로서 서용된 값의 설정값과 동적 SQL문 내의 규정화되지 않은 오브젝트 참조의 내재된 규정화를 위해 사용된 값의 설정값을 나타냅니다. DYNAMICRULES 옵션은 다음 테이블에 설명된 효과를 가지고 있습니다.

표 1. OWNER 및 QUALIFIER에 의해 영향받는 정적 SQL 특성

기능	OWNER만 지정됨	QUALIFIER만 지정됨	QUALIFIER 및 OWNER가 지정됨
사용된 권한 부여 ID	BIND 명령상의 OWNER 옵션에서 지정된 사용자의 ID.	패키지를 바인딩한 사용자의 ID	BIND 명령상의 OWNER 옵션에서 지정된 사용자의 ID.
사용된 규정화되지 않은 규정화 값	BIND 명령상의 OWNER 옵션에서 지정된 사용자의 ID.	BIND 명령상의 QUALIFIER 옵션에서 지정된 사용자의 ID.	BIND 명령상의 QUALIFIER 옵션에서 지정된 사용자의 ID.

표 2. DYNAMICRULES, OWNER 및 QUALIFIER에 영향받는 동적 SQL 특성

기능	RUN	BIND
사용된 권한 부여 ID	패키지를 실행하는 사용자의 ID	패키지에 대한 권한 부여 ID
사용된 규정화되지 않은 규정화 값	CURRENT SCHEMA 특수 레지스터	패키지에 대한 권한 부여 ID

DYNAMICRULES 옵션에 관한 고려사항:

- CURRENT SCHEMA 특수 레지스터는 DYNAMICRULES BIND로 바인드된 패키지로부터 실행된 동적 SQL 상태 내에서 규정화되지 않은 오브젝트 참조를 규정화하는 데 사용되지 않을 것입니다. 대신, DB2는 테이블에서 보여주는 바와 같이 패키지 기본 규정자를 사용할 것입니다. 이는 CURRENT SCHEMA 특수 레지스터를 변경하기 위해 SET CURRENT SCHEMA문을 발행한 후에도 해당됩니다. 레지스터 값은 변경되나 사용되지는 않을 것입니다.
- 다음과 같이 동적으로 준비된 SQL문은 DYNAMICRULES BIND 옵션인 GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT ON, RENAME, SET INTEGRITY, SET EVENT MONITOR STATE 및 별명을 참조하는 조회로 바인드된 패키지 내에 사용될 수 없습니다.
- 단일 연결 동안 여러 개의 패키지가 참조되면, 명령문이 바인드되는 패키지에 대한 BIND 옵션에 따라 동적 SQL이 행동할 것입니다.
- DYNAMICRULES BIND로 바인드할 때, 동적 명령문이 패키지 사용자의 권한 부여 ID를 사용하고 있을 것이므로 패키지의 바인더는 패키지의 사용자에게 주고 싶은 어떠한 권한도 가져선 안된다는 것을 명심하십시오.

권한 부여 ID 및 명령문 준비

BIND에서 VALIDATE BIND를 지정할 경우, 테이블과 뷰를 조작하기 위해 필요한 특권은 바인드할 때 존재해야 합니다. 특권이나 참조된 오브젝트가 존재하지 않고 SQLERROR NOPACKAGE가 적용될 경우, 바인드 조작은 실패합니다. SQLERROR CONTINUE를 지정할 경우, 바인드는 성공적으로 수행되고 오류가 있는 명령문은 플래그 처리됩니다. 오류로 플래그 처리된 명령문을 실행하려고 하면 응용프로그램에서 오류가 발생합니다.

패키지가 VALIDATE RUN으로 바인될 경우, 모든 정상적인 BIND 처리는 완료되지만, 응용프로그램에서 참조되는 테이블과 뷰를 사용하기 위해 필요한 특권은 이 때 존재하지 않아도 됩니다. 명령문에 대해 필요한 특권이 바인드 시 존재하지 않을 경우, 증분식 바인드는 응용프로그램 내에서 명령문이 처음 실행될 때마다 수행되고, 그 명령문에 대해 필요한 모든 특권이 존재해야 합니다. 특권이 존재하지 않을 경우, 명령문 실행은 실패합니다. 런타임에서 권한 부여 점검이 수행될 때, 이것은 패키지 소유자의 권한 부여 ID를 사용하여 수행됩니다.

데이터 유형

컬럼의 데이터 유형 지정에 대해서는, 800 페이지의 『CREATE TABLE』에서 자세한 내용을 참조하십시오.

SQL에서 조작할 수 있는 가장 작은 단위는 *값(value)*입니다. 값을 해석하는 방법은 이들 소스의 데이터 유형에 따라 다릅니다. 값의 소스는 다음과 같습니다.

- 상수
- 컬럼
- 호스트 변수
- 함수
- 표현식
- 특수 레지스터

DB2는 다수의 내장형 데이터 유형을 지원합니다. 이에 대해서는 여기서 설명됩니다. 이는 또한, 사용자 정의 데이터 유형도 지원합니다. 사용자 정의 데이터 유형에 대한 101 페이지의 『사용자 정의 유형』에서 자세한 내용을 참조하십시오.

그림9에는 지원되는 내장형 데이터 유형이 나와 있습니다.

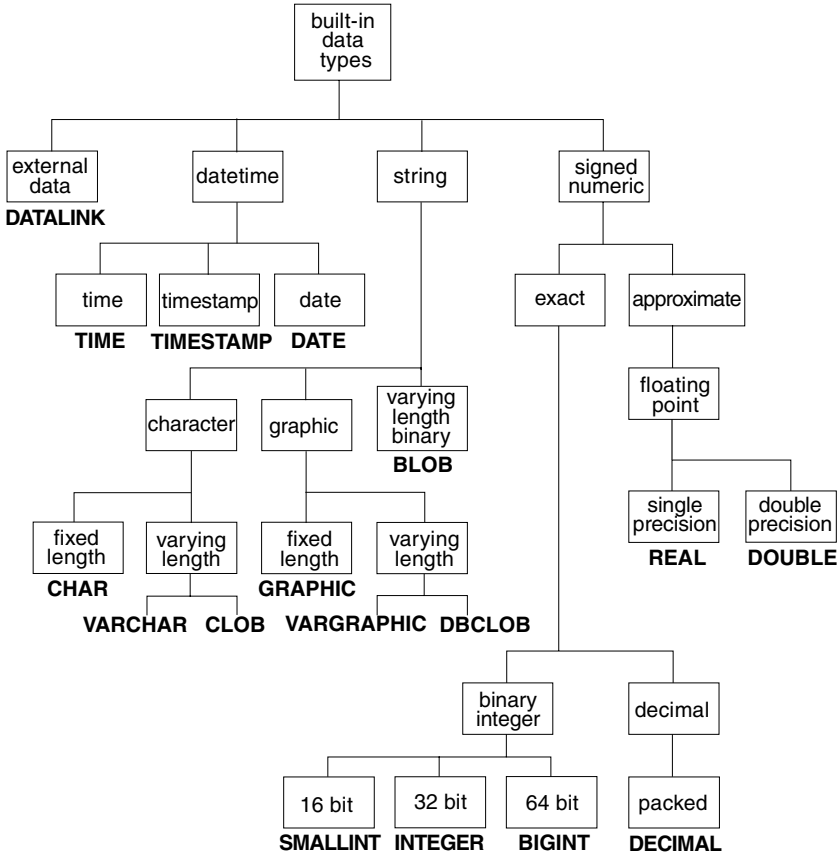


그림 9. 지원되는 내장형 데이터 유형

널(NULL)

모든 데이터 유형에는 널(NULL)값이 포함됩니다. 널(NULL)값은 모든 널(NULL)이 아닌 값과 구별되는 특수값으로, (널(NULL)이 아닌)값이 없음을 나타냅니다. 모든 데이터 유형에 널(NULL) 값이 포함되더라도, NOT NULL로 정의된 컬럼은 널(NULL) 값을 가질 수 없습니다.

대형 오브젝트(LOB)

*대형 오브젝트(LOB)*란 용어와 일반적인 약어 *LOB*는 BLOB, CLOB 또는 DBCLOB 데이터 유형을 참조하는 데 사용됩니다. LOB 값은 92 페이지의 『가변 길이 문자열 사용시 제한사항』에 지정된 LONG VARCHAR 값에 적용되는 제한을 받습니다. LOB 문자열의 경우, 이러한 제한사항은 문자열의 길이 속성이 254바이트이거나 짧을 때에도 적용됩니다.

문자 대형 오브젝트(CLOB) 문자열

*문자 대형 오브젝트(CLOB)*는 그 길이를 최대 2 기가바이트(2 147 483 647바이트)까지 사용할 수 있는 바이트 단위의 가변 길이 문자열입니다. *CLOB*는 단일 문자 세트로 기록된 문서와 같은 대형 SBCS 또는 혼합(SBCS 및 MBCS) 문자 기본 데이터를 저장하는 데 사용됩니다(그러므로, 이와 연관된 SBCS 또는 혼합 코드 페이지가 있습니다). CLOB를 문자열로 간주한다는 점에 유의하십시오.

2바이트 문자 대형 오브젝트(DBCLOB) 문자열

*2바이트 문자 대형 오브젝트(DBCLOB)*는 길이가 1 073 741 823자까지 가능할 수 있는 2 바이트 문자의 가변 길이 문자열입니다. *DBCLOB*는 단일 문자 세트로 기록된 문서같은 대형 DBCS 문자 기본 데이터를 저장하는 데 사용됩니다(그러므로, 이와 연관된 DBCS CCSID가 있습니다.). DBCLOB를 그래픽 문자열로 간주함을 기억하십시오.

2진 대형 오브젝트(BLOB)

*2진 대형 오브젝트(BLOB)*는 그 길이를 최대 2기가바이트(2 147 483 647바이트)까지 사용할 수 있는 바이트 단위의 가변 길이 문자열입니다. *BLOB*는 주로 그림이나, 음성 또는 멀티 미디어 같은 비정규적인 데이터를 보관하기 위한 것입니다. 또 다른 사용은 사용자 정의 유형과 사용자 정의 함수에 이용하기 위해 구조화된 데이터를 보관하는 것입니다. FOR BIT DATA 문자열에서와 마찬가지로, BLOB 문자열은 문자 집합과 관련이 없습니다.

위치 지정자가 있는 대형 오브젝트(LOB) 조작

LOB 값은 매우 클 수 있으므로, 이러한 값을 데이터베이스 서버에서 클라이언트 응용프로그램 호스트 변수로 전송하는 작업에는 시간이 소요될 수 있습니다. 그러나, 응용프로그램은 한번에 전체가 아닌 하나의 LOB 값을 처리하기도 합니다. 응

용프로그램이 전체 LOB를 응용프로그램 메모리에 보관하지 않아도 되는 경우, 응용프로그램은 대형 오브젝트 위치 지정자(LOB 위치 지정자)를 사용하여 LOB 값을 참조할 수 있습니다.

*대형 오브젝트(LOB) 위치 지정자*나 LOB 위치 지정자는 데이터베이스 서버에서 단일 LOB 값을 나타내는 값을 갖는 호스트 변수입니다. LOB 위치 지정자는 응용프로그램이 수행되고 있는 클라이언트 머신에 전체 LOB 값을 보관하지 않고도 응용프로그램에서 대형 오브젝트(LOB)를 쉽게 조작할 수 있는 기법을 사용자에게 제공하도록 개발되었습니다.

예를 들어, LOB 값을 선택하면, 응용프로그램이 전체 LOB 값을 선택하여 이것을 같은 대형 호스트 변수(응용프로그램이 한번에 LOB 값 전체를 처리하려는 경우에)로 옮기거나 대신 LOB 값을 LOB 위치 지정자로 선택하도록 할 수 있습니다. 그런 다음, LOB 위치 지정자를 사용하여 입력으로 위치 지정자를 제공하여 응용프로그램은 LOB 값에 대해 연속적인 데이터베이스 조작을 수행할 수 있습니다(스칼라 함수 SUBSTR, CONCAT, VALUE, LENGTH, 지정 LIKE나 POSSTR의 LOB 검색 또는 LOB에 대한 UDF 적용). 클라이언트 호스트 변수에 데이터량이 지정된 경우, 위치 지정자 조작의 결과는 입력 LOB 값들의 작은 부속 집합입니다.

LOB 위치 지정자는 또한 기본값 이상의 것을 의미합니다. 이들은 LOB 표현식과 연관된 값을 나타내기도 합니다. 예를 들면, LOB 위치 지정자는 다음과 연관된 값을 나타낼 수도 있습니다.

```
SUBSTR( <lob 1> CONCAT <lob 2> CONCAT <lob 3>, <start>, <length> )
```

응용프로그램에 있는 일반 호스트 변수의 경우, 널(NULL)값이 호스트 변수에 대해 선택되면, 표시기 변수는 -1로 설정되어 그 값이 널(NULL)임을 나타냅니다. 그러나 LOB 위치 지정자의 경우, 표시기 변수의 의미는 약간 다릅니다. 위치 지정자 호스트 변수 자체가 널(NULL)이 될 수는 없으므로, 음의 표시기 변수 값은 LOB 위치 지정자가 나타내는 LOB 값이 널(NULL)임을 나타냅니다. 널(NULL) 정보는 위치 지정자 변수 값에 의해 클라이언트에 대해서 지역 정보로 보존됩니다 -- 서버는 유효한 위치 지정자로 널(NULL) 값을 추적하지 않습니다.

LOB 위치 지정자가 데이터베이스에 있는 행이나 위치가 아닌 값을 나타내고 있다는 것을 이해하는 것이 중요합니다. 일단 위치 지정자가 선택되면, 위치 지정자가 참조하는 값이 영향을 주는 원래의 행이나 행에 아무런 조작을 수행할 수 없습니다. 위치 지정자와 연관된 값은 트랜잭션이 종료되거나 위치 지정자가 명확히 해제될 때까지(어느 것이 먼저 일어나도 상관없음) 유효합니다. 위치 지정자는 이 함수를 제공하기 위해 데이터를 여분으로 복사하도록 하지 않습니다. 대신, 위치 지정자 기법은 기본 LOB 값의 설명을 저장합니다. LOB 값(또는, 위의 표시된 것과 같은 표현식)을 구체화하는 것은 실제로 일부 위치에 지정될 때까지 지연됩니다. 호스트 변수의 양식으로 사용자 버퍼에 지정되거나, 데이터베이스에서 또 다른 레코드의 필드 값에 지정됩니다.

LOB 위치 지정자는 트랜잭션동안 LOB 값을 참조하는 데 사용되는 유일한 기법입니다. 이것은 작성된 트랜잭션의 범위를 벗어나지 않습니다. 또한 이것은 데이터베이스 유형이 아니며, 데이터베이스에 저장될 수 없습니다. 결과적으로 뷰나 점검 제한조건에 참여할 수 없습니다. 그러나, 위치 지정자가 LOB 유형의 클라이언트 표현이므로, FETCH, OPEN 및 EXECUTE문에 의해 사용되는 SQLDA 구조 내에서 설명될 수 있는 LOB 위치 지정자에 대한 SQLTYPE입니다.

문자열

*문자열*은 일련의 바이트입니다. 문자열의 길이는 문자열에 있는 바이트 수입니다. 길이가 0인 경우, 값을 *빈 문자열*이라고 합니다. 이 값은 널(NULL)값과 혼돈해서는 안 됩니다.

고정 길이 문자열

고정 길이 문자열 컬럼에 있는 모든 값의 길이는 같으며, 이것은 컬럼의 길이 속성에 의해 결정됩니다. 길이 속성은 1 - 254 이내의 값이여야 합니다.

가변 길이 문자열

가변 길이 문자열에는 VARCHAR, LONG VARCHAR 및 CLOB의 세 가지 유형이 있습니다.

- VARCHAR 유형은 최대 32 672바이트의 가변 길이 문자열입니다.
- LONG VARCHAR 유형은 최대 32 700바이트의 가변 길이 문자열입니다.
- CLOB 유형은 최대 2 기가바이트인 가변 길이 문자열입니다.

가변 길이 문자열 사용시 제한사항: 최대 길이가 255 바이트 이상인 가변 길이 문자열 데이터 유형을 야기하는 표현식에 제한사항이 적용됩니다. 그러한 표현식은 다음에서 허용되지 않습니다.

- DISTINCT가 앞에 있는 SELECT 목록
- GROUP BY절
- ORDER BY절
- DISTINCT의 컬럼 함수
- UNION ALL 이외의 집합 연산자 부속 선택

위에 나열된 제한사항 외에도, LONG VARCHAR를 야기하는 표현식, CLOB 데이터 유형 또는 구조화된 유형 컬럼은 다음에서 허용되지 않습니다.

- Basic, Quantified, BETWEEN 또는 IN 술어
- 컬럼 함수
- VARGRAPHIC, TRANSLATE 및 datetime 스칼라 함수
- LIKE 술어에 있는 패턴 피연산자나 POSSTR 함수에 있는 검색 문자열 피연산자
- 날짜 시간 값의 문자열 표현

인수로 VARCHAR를 취하는 SYSFUN 스키마에 있는 함수들은 그 길이가 4 000 바이트보다 큰 VARCHAR를 인수로 받아들이지 않을 것입니다. 그러나, 이들 함수 중 많은 것들이 또한 CLOB(1M)을 받아들이는 대체 서명을 가집니다. 이러한 함수의 경우, 사용자는 4 000개보다 많은 VARCHAR 문자열을 명시적으로 CLOB로 유형변환(cast)한 후 결과를 다시 원하는 길이의 VARCHAR로 다시 유형변환(cast)합니다.

NUL 종료 문자열

C에서의 NUL 종료 문자열은 사전 처리 컴파일 옵션의 표준 레벨에 따라 다르게 처리됩니다. NUL 종료 문자열의 처리에 대해서는 *응용프로그램 개발 안내서*에 있는 C 언어 관련 절을 참조하십시오.

문자 부속유형

각 문자열은 다음 중 하나로 정의됩니다.

비트 데이터 코드 페이지와 관련 없는 데이터.

SBCS 데이터 모든 문자가 1바이트로 표시되는 데이터.

혼합 데이터 1바이트 문자 집합과 복수 바이트 문자 집합(MBCS)의 문자가 혼합될 수 있는 데이터.

SBCS 및 MBCS에 관한 고려사항: SBCS 데이터는 SBCS 데이터베이스에서만 지원됩니다. 혼합 데이터는 MBCS 데이터베이스에서만 지원됩니다.

그래픽 문자열

*그래픽 문자열*은 2바이트 문자 데이터를 나타내는 일련의 바이트입니다. 문자열의 길이는 문자열에 있는 2바이트 문자의 수입니다. 길이가 0인 경우, 값을 빈 문자열이라고 합니다. 이 값은 널(NULL)값과 혼돈해서는 안 됩니다.

그래픽 문자열은 이들 값에 2바이트 문자 코드 값만 들어 있는지 확인하지 않습니다. [15] 오히려, 데이터베이스 관리 프로그램은 2 바이트 문자 데이터가 그래픽 데이터 필드 내에 포함되어 있는 것으로 간주합니다. 데이터베이스 관리 프로그램은 그래픽 문자열값의 길이가 짝수인지 점검합니다.

그래픽 문자열 데이터 유형은 고정 길이이거나 가변 길이입니다. 고정 길이와 가변 길이의 의미는 문자열 데이터 유형에 정의된 것과 비슷합니다.

고정 길이 그래픽 문자열

고정 길이 그래픽 문자열 컬럼에 있는 모든 값의 길이는 같으며, 이것은 컬럼의 길이 속성에 의해 결정됩니다. 길이 속성은 1 - 127 이내여야 합니다.

가변 길이 그래픽 문자열

가변 길이 그래픽 문자열에는 VARGRAPHIC, LONG VARGRAPHIC 및 DBCLOB의 세 가지 유형이 있습니다.

- VARGRAPHIC 유형은 최대 16 336의 2바이트 문자인 가변 길이 문자열입니다.
- LONG VARGRAPHIC 유형은 최대 16 350의 2바이트 문자인 가변 길이 문자열입니다.

15. 이 규칙의 예외는 WCHARTYPE CONVERT 옵션으로 사전 처리 컴파일(Precompile)된 응용프로그램입니다. 이 경우, 유효성 검사가 수행됩니다. *응용프로그램 개발 안내서*의 『C 및 C++ 프로그래밍』에서 자세한 내용을 참조하십시오.

- DBCLOB 유형은 최대 1 073 741 823의 2바이트 문자인 가변 길이 문자열입니다.

특수 제한사항이 최대 길이가 127 이상인 가변 길이 그래픽 문자열로 구성된 표현식에 적용됩니다. 이러한 제한사항은 92 페이지의 『가변 길이 문자열 사용시 제한사항』에서 지정된 것과 같습니다.

NUL 종료 그래픽 문자열

C에서의 NUL 종료 그래픽 문자열은 사전 처리 컴파일 옵션의 표준 레벨에 따라 다르게 처리됩니다. NUL 종료 그래픽 문자열의 처리에 대해서는 *응용프로그램 개발 안내서*에 있는 C 언어 관련 절을 참조하십시오.

이 데이터 유형은 테이블에서 작성될 수 없습니다. 데이터베이스에 데이터를 입력하거나 데이터베이스에서 데이터를 검색하는 데만 사용할 수 있습니다.

2진 문자열

*2진 문자열*은 일련의 바이트입니다. 대개 텍스트 데이터가 들어 있는 문자열과는 달리, 2진 문자열은 그림과 같이 특수한 데이터를 보관하는 데 사용됩니다. '비트 데이터' 부속 유형의 문자열과 유사한 용도로 사용될 수 있으나 두 데이터 유형간에는 호환성이 없습니다. BLOB 스칼라 함수는 비트 문자열의 문자를 2진 문자열로 유형변환(cast)하는 데 사용할 수 있습니다. 2진 문자열의 길이는 바이트 수입니다. 이것은 코드 페이지와 연관되지 않습니다. 2진 문자열에는 문자열과 같은 제한사항(92 페이지의 『가변 길이 문자열 사용시 제한사항』 참조)이 적용됩니다.

숫자

모든 숫자에는 *부호*와 *정밀도*가 있습니다. *정밀도*는 부호를 제외한 비트 수 또는 디지트 수입니다. 부호는 숫자가 0이면 양수로 가정됩니다.

작은 정수(SMALLINT)

*작은 정수*는 5 디지트의 정밀도를 갖는 2바이트 정수입니다. 작은 정수의 범위는 -32 768에서 32 767까지입니다.

큰 정수(INTEGER)

*큰 정수*는 10 디지트의 정밀도를 갖는 4바이트 정수입니다. 큰 정수의 범위는 -2 147 483 648에서 +2 147 483 647까지입니다.

대 정수(BIGINT)

*대 정수*는 19 디지트의 정밀도를 가진 8바이트의 정수입니다. 대 정수의 점위는 -9 223 372 036 854 775 808에서 +9 223 372 036 854 775 807까지입니다.

단일 정밀도 부동 소수점(REAL)

단일 정밀도 부동 소수점 수는 대략 32 비트의 실수입니다. 이 수는 0이거나 -3.402E+38에서 -1.175E-37까지 또는 1.175E-37에서 3.402E+38까지를 범위로 합니다.

배밀도 부동 소수점(DOUBLE 또는 FLOAT)

배밀도 부동 소수점 수는 대략 64 비트의 실수입니다. 수는 0이나 -1.79769E+308에서 -2.225E-307 또는 2.225E-307에서 1.79769E+308까지의 범위입니다.

소수(십진수 또는 NUMERIC)

소수값은 내재 소수점이 있는 팩된 십진수 값입니다. 소수점의 위치는 정밀도와 스케일에 의해 결정됩니다. 수의 소수점 이하 디지트의 수인 스케일은 음수이거나 정밀도보다 클 수 없습니다. 최대 정밀도는 31 디지트입니다. 팩된 소수 표현에 대해서는 1279 페이지의 『팩된(packed) 십진수』에서 참조하십시오.

소수 컬럼의 모든 값의 정밀도와 스케일은 같습니다. 십진수 컬럼에서 십진 변수나 숫자의 범위는 *-n*에서 *+n*까지입니다. *n*의 절대값은 적용 가능한 정밀도와 스케일로 표시될 수 있는 가장 큰 수입니다. 최대 범위는 -10**31+1에서 10**31-1까지입니다.

날짜 시간 값

날짜 시간 데이터 유형은 아래에서 설명됩니다. 날짜 시간값은 어떤 연산이나 문자열 조작에서 사용할 수 있고 어떤 문자열과 호환된다 하더라도, 이들은 문자열이나 수가 아닙니다.

날짜

*날짜*는 세 부분 값(년, 월, 일)으로 구성됩니다. 년 부분 값의 범위는 0001 - 9999입니다. 월 부분값의 범위는 1- 12입니다. 일 부분 값의 범위는 1 - *x*까지이며, 여기서 *x*는 월에 따라 다릅니다.

날짜의 내부 표현은 4바이트의 문자열입니다. 각 바이트는 2개의 팩된 소수 디지트로 구성됩니다. 처음 2바이트는 년을, 그 다음 셋째 바이트는 월을, 마지막 바이트는 일을 나타냅니다.

SQLDA에서 설명한 바와 같이, DATE 컬럼의 길이는 10바이트이며, 이것은 날짜값의 문자열 표현에 적합한 길이입니다.

시간

*시간*은 24시간 하루를 나타내는 세 부분의 값(시, 분, 초)으로 구성됩니다. 시간값의 범위는 0-24이며, 다른 것의 범위는 0-59입니다. 시간이 24인 경우, 분과 초 스펙은 0이 됩니다.

시간의 내부 표현은 3바이트의 문자열입니다. 각 바이트는 2개의 팩된 십진수 디지트입니다. 첫번째 바이트는 시간을, 그 다음 바이트는 분을, 마지막 바이트는 초를 나타냅니다.

SQLDA에서 설명한 바와 같이, TIME 컬럼의 길이는 8바이트이며, 이것은 값의 문자열 표현에 적합한 길이입니다.

시각

*시각*은 위에서 설명한 날짜와 시간을 나타내는 7 부분(년, 월, 일, 시, 초, 분 및 마이크로초)으로 구성되며, 이때 마이크로초의 분수 부분이 포함되는 것만 제외됩니다.

시각의 내부 표현은 10바이트 문자열이며, 각각은 2개의 팩된 소수 디지트로 구성됩니다. 처음 4바이트는 일을, 그 다음 3바이트는 시를, 마지막 3바이트는 마이크로초를 나타냅니다.

SQLDA에서 설명한 바와 같이, TIMESTAMP 컬럼의 길이는 26바이트이며, 이것은 값의 문자열 표현에 적합한 길이입니다.

날짜 시간값의 문자열 표현

데이터 유형이 DATE, TIME 또는 TIMESTAMP인 값은 SQL 사용자가 알 수 없는 내부 양식으로 표현됩니다. 그러나 날짜, 시간 및 시각은 문자열로도 표현되며, 이러한 표현은 데이터유형이 DATE, TIME 또는 TIMESTAMP인 상수나 변수가 없으므로 SQL 사용자에게 직접 영향을 줍니다. 그러므로, 날짜 시간 값을 검

색하려면, 문자열 변수에 날짜 시간 값을 할당해야 합니다. CHAR 함수는 날짜 시간 값을 문자열 표현으로 변경하는 데 사용될 수 있음을 기억하십시오. 프로그램이 데이터베이스에 대해 사전 처리 컴파일되고 바인드될 때 *DATETIME* 옵션 스펙으로 대체되지 않는 한, 문자열 표현은 데이터베이스의 국가 코드와 연관된 날짜 시간값의 기본 형식이 됩니다.

길이에 관계없이, 대형 오브젝트(LOB) 문자열이나 LONG VARCHAR은 날짜 시간 값을 나타내는 문자열로 사용할 수 없습니다. 사용하는 경우 오류가 발생합니다(SQLSTATE 42884).

날짜 시간값의 유효한 문자열 표현이 내부 날짜 시간값을 이용한 수행에서 사용되면, 문자열 표현은 수행되기 전에 날짜, 시간 또는 시각의 내부 형식으로 변환됩니다. 다음 절에서는 날짜 시간값의 유효한 문자열 표현에 대해 정의합니다.

날짜 문자열

날짜의 문자열 표현은 시작하는 하나의 디지트로 길이가 최소 8자인 문자열입니다. 뒤에 붙은 공백도 포함됩니다. 월과 일 위치에서 앞에 있는 0은 생략 가능합니다.

날짜의 유효한 문자열 형식은 표 1에 나열되어 있습니다. 각 형식은 이름으로 식별되며 연관된 약어와 사용 예가 포함되어 있습니다.

표 3. 날짜 문자열 형식

형식 이름	축약어	날짜 형식	예
국제 표준화 기구	ISO	yyyy-mm-dd	1991-10-27
IBM USA 표준	USA	mm/dd/yyyy	10/27/1991
IBM 유럽 표준	EUR	dd.mm.yyyy	27.10.1991
일본 산업 규격	JIS	yyyy-mm-dd	1991-10-27
사이트 *DB2 Data Links Manager 빠른 시작*	LOC	데이터베이스 국가 코드에 따라 다름.	--

시간 문자열

시간 문자열 표현은 숫자로 시작하는 길이가 최소 4자인 문자열입니다. 뒤에 오는 공백은 포함되지만, 시간의 시 부분에서 앞에 오는 0은 생략 가능하며, 초는 전체가 생략될 수 있습니다. 초를 생략하면, 0 초의 암시적인 표현으로 간주합니다. 그러므로, 13.30은 13.30.00과 같습니다.

시간의 유효한 문자열 형식은 표4에 나와 있습니다. 각 형식은 이름으로 식별하며, 연관된 축약어와 사용 예가 포함됩니다.

표 4. 시간 문자열 표현 형식

형식 이름	축약어	시간 형식	예
국제 표준화 기구[2]	ISO	hh.mm.ss	13.30.05
IBM USA 표준	USA	hh:mm AM 또는 PM	1:30 PM
IBM 유럽 표준	EUR	hh.mm.ss	13.30.05
일본 산업 규격	JIS	hh:mm:ss	13:30:05
사이트 *DB2 Data Links Manager 빠른 시작*	LOC	데이터베이스 국가 코드에 따라 다름.	--

주:

1. ISO에서, EUR 및 JIS 형식, .ss(또는 :ss)는 선택적입니다.
2. 국제 표준화 기구는 최근 일본 산업 규격과 같게 시간 형식을 변경했습니다. 그러므로, 응용프로그램이 현재의 국제 표준화 기구 형식을 사용하는 경우 JIS 형식을 사용하십시오.
3. USA 시간 문자열 형식의 경우, 분은 생략할 수 있으며, 이것은 00분의 내재적 표현으로 간주됩니다. 그러므로, 1 PM은 1:00 PM과 같습니다.
4. USA 시간 형식에서, 시간은 12보다 클 수 없으며 00:00 AM과 같은 특수 경우를 제외하고는 0이 될 수 없습니다. AM과 PM 앞에는 하나의 공백이 있습니다. 24시간의 ISO 형식을 사용하는, USA 형식과 24시간 형식과의 대응은 다음과 같습니다.

   12:01 AM - 12:59 AM은 00.01.00 - 00.59.00에 해당됩니다.
   01:00 AM - 11:59 AM은 01.00.00 - 11.59.00에 해당됩니다.
   12:00 PM(정오) - 11:59 PM은 12.00.00 - 23.59.00에 해당됩니다.
   12:00 AM(자정)은 24.00.00에 해당되고 00:00 AM(자정)은 00.00.00에 해당됩니다.

시간소인 문자열

시간소인의 문자열 표현은 하나의 디지트로 시작하여 길이가 최소 16자인 문자열입니다. 시간소인의 완전한 문자열 표현은 *yyyy-mm-dd-hh.mm.ss.nnnnnn*양식입니다. 뒤에 오는 공백은 포함됩니다. 앞에 오는 0은 시각의 월, 일, 시 부분에서 생

략될 수 있고, 마이크로초는 절단하거나 전체를 생략할 수 있습니다. 뒤에 오는 0 디지트가 마이크로초 부분에서 생략되면, 생략된 디지트의 내재적 표현인 것으로 간주합니다. 그러므로, 1991-3-2-8.30.00은 1991-03-02-08.30.00.000000과 같습니다.

SQL문은 ODBC 문자열 형식의 시각을 입력 값으로만 지원합니다. ODBC 문자열 형식의 시각은 *yyyy-mm-dd hh:mm:ss.nnnnnn*의 형식을 취합니다. ODBC에 대한 *CLI Guide and Reference*에서 자세한 정보를 참조하십시오.

MBCS에 관한 고려사항

날짜, 시간 및 시각 문자열에는 1바이트 문자 및 디지트만 포함될 수 있습니다.

DATALINK 값

DATALINK 값은 데이터베이스로부터 데이터베이스 외부에 저장된 파일에 이르기까지 논리 참조가 포함된 캡슐화된 값입니다. 이 캡슐화된 값의 속성은 다음과 같습니다.

링크 유형

현재 지원되는 링크 유형은 'URL'(Uniform Resource Locator)입니다.

데이터 위치

URL 양식에서 DB2 내에 참조와 링크된 파일의 위치. 이 URL에 대해 허용되는 스킴 이름은 다음과 같습니다.

- HTTP
- FILE
- UNC
- DFS

URL의 다른 부분은 다음과 같습니다.

- HTTP, FILE, UNC 스킴의 파일 서버 이름
- DFS 스킴에 대한 셀 이름
- 파일 서버나 셀 내에서의 전체 파일 경로 이름

DATALINK에 대한 정확한 BNF(Backus Naur form) 스펙에 대해서는 15 15 페이지의『부록P. DATALINK를 위한 BNF 스펙』에서 자세한 정보를 참조하십시오.

주석

최고 254바이트의 설명 정보. 이는 데이터 위치 식별과 같이, 응용프로그램의 특정 용도를 위한 것입니다.

앞 및 뒤 공백 문자는 데이터 위치 속성을 URL로 분석하는 동안 잘립니다. 또한, 스킴 이름('http', 'file', 'unc', 'dfs')과 호스트는 대소문자가 구별되며 항상 데이터베이스에 대문자로 저장됩니다. DATALINK 값이 데이터베이스에서 페치된 경우, 액세스 토큰은 적절할 때 URL 속성 내에 embed됩니다. 이것은 동적으로 생성되고 데이터베이스에 저장된 DATALINK 값의 영구 부분이 아닙니다. 세부사항은 321 페이지의『DLCOMMENT』에서부터 DATALINK 관련 스칼라 함수를 참조하십시오.

DATALINK 값이 주속 속성과 빈 데이터 위치 속성만 갖도록 할 수 있습니다. 이런 값도 컬럼에 저장될 수는 있지만, 이런 컬럼에는 파일을 연결할 수 없습니다. 주석과 DATALINK 값의 데이터 위치 속성의 총 길이는 현재 200 바이트로 제한됩니다.

DATALINK는 DRDA 서버와 교환될 수 없습니다.

159 페이지의『BLOB, CLOB 및 DBCLOB 호스트 변수 참조』절에 설명된 LOB 파일 참조 변수와 파일에 대한 이러한 DATALINK 참조를 구별하는 것은 중요합니다. 두 참조 모두 파일 표시가 들어 있으므로 유사해 보입니다. 그러나, 다음과 같은 점에서 구별됩니다.

- DATALINK는 데이터베이스에 들어 있고, 링크와 링크된 파일에 있는 데이터 모두가 데이터베이스에 있는 데이터가 자연스럽게 확장된 것으로 간주될 수 있습니다.
- 파일 참조 변수는 일시적으로 클라이언트에 존재하며, 이를 호스트 프로그램 버퍼 대용으로 할 수 있습니다.

내장 스칼라 함수가 제공되어 DATALINK 값(DLVALUE)을 구축하고, DATALINK 값으로부터 캡슐화된 값들(DLCOMMENT, DLLINKTYPE, DLURLCOMPLETE, DLURLPATH, DLURLPATHONLY, DLURLSCHEME, DLURLSERVER)을 추출합니다.

사용자 정의 유형

구별 유형

*구별 유형(distinct type)*은 사용자가 정의하는 데이터 유형으로 그 내부 표현과 기존 유형(『소스』 유형)을 공유하지만, 대부분의 조작에서는 별도의 호환성이 없는 유형으로 간주됩니다. 예를 들어, 그림 유형, 텍스트 유형 및 오디오 유형을 정의하려고 하며, 이들 모두는 서로 다른 의미가 있지만 내부 표현으로 내장 데이터 유형인 BLOB를 사용합니다.

다음 예는 AUDIO라는 구별 유형의 작성에 대해 설명합니다.

```
CREATE DISTINCT TYPE AUDIO AS BLOB (1M)
```

AUDIO가 내장 데이터 유형인 BLOB와 동일하게 표현되더라도, BLOB나 그 밖의 유형과 비교할 수 없는 별도의 유형으로 간주됩니다. 그러면, AUDIO에 대해 특별히 작성된 함수를 작성할 수 있으며, 이 함수는 다른 유형(그림, 텍스트 등)에는 적용되지 않습니다.

구별 유형은 규정화된 식별자로 구별됩니다. 스키마 이름이 CREATE DISTINCT TYPE, DROP DISTINCT TYPE 또는 COMMENT ON DISTINCT TYPE문 이외의 명령문에 사용될 경우, 구별 유형 이름을 규정화하는 데 스키마 이름이 사용되지 않으면 *SQL 경로*를 순서대로 검색하여 일치하는 구별 유형을 가진 첫번째 스키마를 찾습니다. SQL 경로는 142 페이지의 『CURRENT PATH』에서 설명됩니다.

구별 유형은 구별 유형으로 명확하게 정의된 함수와 조작만이 이 인스턴스에 적용될 수 있음을 확실히 하여 강력하게 지원됩니다. 이런 이유로, 구별 유형은 소스 유형의 함수와 연산자를 자동으로 확보할 수 없습니다. 이것은 이들이 의미가 없기 때문입니다(예를 들어, AUDIO의 LENGTH 함수는 바이트보다는 오브젝트의 길이(초 단위)를 반환합니다.).

LONG VARCHAR, LONG VARGRAPHIC, LOB 유형 또는 DATALINK가 소스인 구별 유형은 소스 유형과 같은 제한을 받게 됩니다.

그러나, 소스 유형의 특정 함수와 연산자를 명시적으로 지정하여, 구별 유형의 소스 유형에 정의된 함수에 기초한 사용자 정의 함수를 정의하여 구별 유형에 적용할 수 있습니다(예를 들면, 123 페이지의 『사용자 정의 유형 비교』 참조). 비교 연산자는 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB 또는 DATALINK를 소스 유형으로 사용하는 구별 유형을 제외하고, 사용자 정의 구별 유형에 대해 자동 생성됩니다. 이 외에도, 소스 유형에서 구별 유형으로, 구별 유형에서 소스 유형으로의 전환을 지원하는 함수가 작성됩니다.

구조화 유형

*구조화 유형*은 데이터베이스에 정의된 구조를 가진 사용자 정의 데이터 유형입니다. 여기에는 *속성*이란 연속이 포함되며, 각각의 속성에는 데이터 유형이 있습니다. 구조화된 유형에는 메소드 스펙 세트도 포함됩니다.

구조화된 유형은 테이블, 뷰 또는 컬럼 유형으로 사용될 수 있습니다. 테이블 또는 뷰에 대한 유형으로 사용될 경우, 테이블이나 뷰는 각각 *입력된 테이블* 또는 *입력된 뷰*로 알려집니다. 입력된 테이블 및 입력된 뷰의 경우, 구조화된 유형의 데이터 이름과 속성은 이 입력된 테이블이나 입력된 뷰의 컬럼에 대한 이름과 데이터 유형이 됩니다. 유형화된 테이블 또는 유형화된 뷰의 행들은 구조화 유형 인스턴스의 표시로 볼 수 있습니다. 컬럼에 대한 데이터 유형으로 사용될 경우, 컬럼에는 구조화된 유형의 값이나, 아래에 설명된 대로 그 유형의 부속 유형 값이 포함됩니다. 메소드는 구조화된 컬럼 오브젝트 속성을 검색하거나 조작하기 위해 사용됩니다.

용어: *수퍼 유형*은 *부속 유형*이라고 하는 다른 구조화 유형이 정의된 구조화된 유형입니다. 부속 유형은 해당되는 수퍼 유형의 모든 속성과 메소드를 계승하며 추가 속성과 메소드가 정의될 수 있습니다. 공통되는 상위 유형과 관련 있는 구조화 유형 집합을 *유형 계층*이라 하고, 상위 유형이 없는 유형을 그 유형 계층의 *루트 유형*이라고 합니다.

부속 유형이라는 용어는, 사용자가 정의한 구조화 유형과, 유형 계층에서 그 아래에 있는 모든 사용자 정의 구조화 유형에 적용됩니다. 그러므로, 구조화 유형 T의

부속 유형은 계층에서 T와 T 아래에 있는 모든 구조화 유형입니다. 구조화 유형 T의 *적절한 부속 유형*은 유형 계층상 T 아래에 있는 구조화 유형입니다.

유형 계층에서의 순환 유형 정의에 대해 제한사항이 있습니다. 이러한 이유로, 허용되는 순환 정의의 특정 유형을 참조하는 간단한 방법을 개발해야 합니다. 다음과 같은 정의가 사용됩니다.

- *직접 사용*: 다음 중 한 가지만 만족될 경우 유형 **A**는 직접적으로 또 다른 유형 **B**를 사용한다고 말할 수 있습니다.
  1. 유형 **A**에 유형 **B**의 속성이 있습니다.
  2. 유형**B**가 **A**의 부속 유형이거나 **A**의 수퍼 유형입니다.
- *간접 사용*: 다음 중 한 가지가 만족될 경우 유형 **A**는 간접적으로 유형 **B**를 사용한다고 말할 수 있습니다.
  1. 유형 **A**는 직접 유형 **B**를 사용합니다.
  2. 유형 **A**는 유형 **C**를 직접 사용하고, 유형 **C**는 간접적으로 유형 **B**를 사용합니다.

유형은 해당되는 속성 유형 중 하나가 직접 또는 간접으로 자체를 사용하도록 정의될 수 없습니다. 그러한 구성이 필요할 경우, 속성으로 참조를 사용하는 것을 고려해 보십시오. 예를 들어, 구조화된 유형 속성에서, ″manager″가 ″employee″ 유형일 경우, 속성이 ″manager″인 ″employee″ 인스턴스가 될 수 없습니다. 그러나, 유형이 REF(employee)인 ″manager″ 속성은 될 수 있습니다.

유형은 특정의 다른 오브젝트에서 그 유형을 직접 또는 간접적으로 사용할 경우 삭제할 수 없습니다. 예를 들어, 유형은 테이블이나 뷰 컬럼이 직접 또는 간접적으로 그 유형을 사용할 경우 삭제할 수 없습니다. 유형 삭제를 제한할 수 있는 모든 오브젝트에 대해 1004 페이지의 표27의 자세한 내용을 참조하십시오.

구조화된 유형 컬럼 값은 92 페이지의 『가변 길이 문자열 사용시 제한사항』에 지정된 CLOB 값에 적용되는 제한을 받습니다.

참조(REF) 유형

*참조 유형*은 구조화 유형의 동료 유형입니다. 구별 유형과 마찬가지로 참조 유형은 내장된 데이터 유형 중 하나와 공통 표시를 공유하는 스칼라 유형을 말합니다. 이러한 동일한 표시는 유형 계층의 모든 유형들이 공유합니다. 참조 유형 표시는

유형 계층의 루트 유형을 작성할 때 정의합니다. 참조 유형을 사용할 때, 구조화 유형은 그 유형의 매개변수로서 지정됩니다. 이 매개변수를 참조의 목표 *유형*이라고 합니다.

참조 목표는 항상 분류된 테이블 또는 뷰에 있는 행입니다. 참조 유형을 사용할 때 *영역*을 정의할 수 있습니다. 영역은 참조 값의 목표 행이 들어 있는 테이블 (또는 *목표 테이블*) 또는 뷰 (또는 *목표 뷰*)를 식별합니다. 목표 테이블 또는 뷰에는 참조 유형의 목표 유형과 동일한 유형이 있어야 합니다. 영역 분류된 참조 유형 인스턴스는 *목표 행*이라는 분류된 뷰 또는 분류된 테이블에 있는 행을 고유하게 식별합니다.

데이터 유형의 승격

데이터 유형은 관련 데이터 유형 그룹으로 분류됩니다. 그룹 내에서, 우선순위가 있으므로 하나의 데이터 유형은 다른 데이터 유형에 선행되는 것으로 간주됩니다. 이러한 우선순위에서 데이터 유형을 뒤에 나오는 데이터 유형으로 승격시킬 수 있습니다. 예를 들어, 데이터 유형 CHAR은 VARCHAR로 승격될 수 있고, INTEGER는 DOUBLE PRECISION으로 승격될 수 있지만, CLOB는 VARCHAR로 승격될 수 없습니다.

데이터 유형의 승격은 다음 경우에 사용됩니다.

- 함수 차수 수행(168 페이지의 『함수 차수』 참조)
- 사용자 정의 유형변환(106 페이지의 『데이터 유형간의 변환』 참조)
- 사용자 데이터 유형을 내장된 데이터 유형에 지정(117 페이지의 『사용자 정의 유형 지정』 참조)

표5는 각 데이터 유형에 대한 순서 목록(순서)을 표시하며, 데이터 유형이 승격될 수 있는 다음 데이터 유형을 결정하는 데 사용합니다. 표에서는 다른 데이터 유형으로 승격하는 대신 언제나 같은 데이터 유형을 사용하는 것이 가장 좋은 선택임을 보여 줍니다.

표 5. 데이터 유형 순서 표

데이터 유형	데이터 유형 순서 목록(좋은것-나쁜것 순)
CHAR	CHAR, VARCHAR, LONG VARCHAR, CLOB

표 5. 데이터 유형 순서 표 (계속)

데이터 유형	데이터 유형 순서 목록(좋은것-나쁜것 순)
VARCHAR	VARCHAR, LONG VARCHAR, CLOB
L O N G VARCHAR	LONG VARCHAR, CLOB
GRAPHIC	GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB
VARGRAPHIC	VARGRAPHIC, LONG VARGRAPHIC, DBCLOB
L O N G VARGRAPHIC	LONG VARGRAPHIC, DBCLOB
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB
SMALLINT	SMALLINT, INTEGER, BIGINT, decimal, real, double
INTEGER	INTEGER, BIGINT, decimal, real, double
BIGINT	BIGINT, decimal, real, double
decimal	decimal, real, double
real	real, double
double	double
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
DATALINK	DATALINK
udt	udt(같은 이름) 또는 udt의 수퍼 유형
REF(T)	REF(S)(S는 T의 상위 유형)

주:

소문자로 된 유형은 다음과 같이 정의됩니다.

십진수 = decimal(p,s) 또는 NUMERIC(p,s)

실수 = REAL 또는 FLOAT(*n*). 여기서 *n*은 24보다 크지 않습니다.

2배수 = DOUBLE, DOUBLE PRECISION, FLOAT 또는 FLOAT(*n*). 여기서 *n*은 24보다 크지 않습니다.

udt = 사용자 정의 유형

열거된 데이터 유형의 짧거나 긴 양식 동의어는 열거된 동의어와 같은 것으로 간주됩니다.

데이터 유형간의 변환

주어진 데이터 유형의 값을 다른 데이터 유형으로 또는 길이, 정밀도 또는 스케일이 다른 같은 데이터 유형으로 *변환*해야 하는 경우도 많습니다. 데이터 유형 승격(104 페이지의 『데이터 유형의 승격』에서 정의한 것처럼)은 하나의 예로, 한 데이터 유형을 다른 데이터 유형으로 승격한다는 것은 값이 새로운 데이터 유형으로 변환된다는 것입니다. 다른 데이터 유형으로 변환할 수 있는 데이터 유형을 소스 데이터 유형에서 목표 데이터 유형으로 *변환 가능*하다고 합니다.

데이터 유형간의 변환은 CAST 스펙(201 페이지의 『유형변환(CAST) 스펙』 참조)을 사용하여 명시적으로 수행할 수 있지만, 사용자 정의 유형이 관련된 지정(117 페이지의 『사용자 정의 유형 지정』 참조)시 내재적으로 시행될 수도 있습니다. 또한 소스 사용자 정의 함수(653 페이지의 『CREATE FUNCTION』 참조)를 작성할 때, 소스 함수 매개변수의 데이터 유형은 작성되고 있는 함수의 데이터 유형으로 변환되어야 합니다.

내장 데이터 유형간에 지원되는 변환은 108 페이지의 표6에 나와 있습니다.

구별 유형이 관련된 다음과 같은 변환이 지원됩니다.

- 구별 유형 *DT*에서 소스 데이터 유형 *S*로 변환.
- 구별 유형 *DT*의 소스 데이터 유형 *S*에서 구별 유형 *DT*로 변환.
- 구별 유형 *DT*에서 같은 구별 유형 *DT*로 변환.
- 데이터 유형 *A*에서 구별 유형 *DT*로의 변환. 여기서 *A*는 구별 유형 *DT*의 소스 데이터 유형 *S*로 승격할 수 있습니다(104 페이지의 『데이터 유형의 승격』 참조).
- INTEGER에서 소스 데이터 유형인 SMALLINT인 구별 유형 *DT*로 변환
- DOUBLE에서 소스 데이터 유형이 REAL인 구별 유형 *DT*로 변환
- VARCHAR에서 소스 데이터 유형이 *DT*인 구별 유형으로 변환
- VARGRAPHIC에서 소스 데이터 유형이 GRAPHIC인 구별 유형 *DT*로 변환

구조화된 유형 값은 다른 어떤 것으로도 유형변환(cast)할 수 없습니다. 구조화된 유형 *ST*는 해당되는 수퍼 유형 중 하나로 유형변환(cast)할 수 없습니다. *ST*의 수퍼 유형에 대한 모든 메소드가 *ST*에만 적용되지 때문입니다. 원하는 조작이 유일

하게 *ST* 부속 유형에만 적용 가능할 경우, *ST*를 해당되는 부속 유형 중 하나로 처리하도록 부속 유형 처리 표현식을 사용하십시오. 214 페이지의 『부속 유형 처리』에서 자세한 내용을 참조하십시오.

변환에 관련된 사용자 정의 데이터 유형에 스키마 이름이 규정되지 않은 경우, 그 이름의 사용자 정의 데이터 유형을 포함하는 첫번째 스키마를 찾는 데 *SQL 경로*가 사용됩니다. SQL 경로는 142 페이지의 『CURRENT PATH』에서 자세히 설명됩니다.

참조 유형과 관련하여 다음과 같은 유형변환(cast)이 지원됩니다.

- 참조 유형 *RT*로부터 표시 데이터 유형 *S*로의 유형변환(cast)
- 참조 유형 *RT*의 표시 데이터 유형 *S*로부터 참조 유형 *RT*로의 유형변환(cast)
- 목표 유형이 *T*인 참조 유형 *RT*로부터 목표 유형이 *S*인 참조 유형 *RS*로의 유형변환(cast). 여기서 *S*는 *T*의 상위 유형입니다.
- 데이터 유형 *A*에서 참조 유형 *RT*로의 유형변환(cast). 여기서 *A*는 참조 유형 *RT*의 표시 데이터 유형 *S*로 승격할 수 있습니다(104 페이지의 『데이터 유형의 승격』 참조).

유형변환(cast)에 관계된 참조 데이터 유형의 목표 유형이 스키마 이름에 의해 규정되지 않은 경우, 그 이름 옆에 사용자 정의 데이터 유형이 포함된 첫번째 스키마를 찾는 데 *SQL 경로*가 사용됩니다. SQL 경로는 142 페이지의 『CURRENT PATH』에서 자세히 설명됩니다.

데이터 유형간의 변환

표 6. 내장 데이터 유형간의 지원되는 변환.

목표 데이터 유형 → / 소스 데이터 유형 ↓	SMALLINT	INTEGER	BIGINT	DECIMAL	REAL	DOUBLE	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARG	DBCLOB	DATE	TIME	TIMESTAMP	BLOB
SMALLINT	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
INTEGER	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
BIGINT	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
DECIMAL	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
REAL	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
DOUBLE	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
CHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	-	Y	-	-	Y	Y	Y	Y
VARCHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	-	Y	-	-	Y	Y	Y	Y
LONG VARCHAR	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	-	-	-	-	Y
CLOB	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	-	-	-	-	Y
GRAPHIC	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y
LONG VARG	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y
DBCLOB	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y
DATE	-	-	-	-	-	-	Y	Y	-	-	-	-	-	-	Y	-	-	-
TIME	-	-	-	-	-	-	Y	Y	-	-	-	-	-	-	-	Y	-	-
TIMESTAMP	-	-	-	-	-	-	Y	Y	-	-	-	-	-	-	Y	Y	Y	-
BLOB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y

주

- 사용자 정의 유형 및 참조 유형과 관련있는 지원되는 유형변환(cast)에 대해서는 테이블 앞의 설명을 참조하십시오.
- DATALINK 유형만 DATALINK 유형으로 변환될 수 있습니다.
- 구조화된 유형 값은 다른 어떤 것으로도 유형변환(cast)할 수 없습니다.

지정 및 비교

SQL의 기본 조작은 지정과 비교입니다. 지정 조작은 INSERT, UPDATE, FETCH, SELECT INTO, VALUES INTO 및 SET 전이 변수 명령문 실행시 수행됩니다. 함수의 인수도 함수 호출시 지정됩니다. 비교 조작은 술어와 MAX, MIN, DISTINCT, GROUP BY 및 ORDER BY와 같은 기타 언어 요소를 포함하는 명령문을 수행하는 동안 수행됩니다.

두 조작을 위한 기본 규칙은 포함된 피연산자의 데이터 유형이 호환되어야 한다는 것입니다. 호환성 규칙은 또한 설정 조작(125 페이지의 『결과 데이터 유형 규칙』 참조)에도 적용됩니다. 호환성 배열은 다음과 같습니다.

표 7. 지정과 비교에 대한 데이터 유형 호환성

피연산자	2진 정수	십진수	부동 소수점	문자열	그래픽 문자열	날짜	시간	시각	2진 문자열	UDT
2진 정수	예	예	예	No	No	No	No	No	No	[2]
십진수	예	예	예	No	No	No	No	No	No	[2]
부동 소수점	예	예	예	No	No	No	No	No	No	[2]
문자열	No	No	No	예	No	[1]	[1]	[1]	아니오 [3]	[2]
그래픽 문자열	No	No	No	No	예	No	No	No	No	[2]
날짜	No	No	No	[1]	No	예	No	No	No	[2]
시간	No	No	No	[1]	No	No	예	No	No	[2]
시각	No	No	No	[1]	No	No	No	예	No	[2]
2진 문자열	No	No	No	아니오 [3]	No	No	No	No	예	[2]
UDT	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	예

표 7. 지정과 비교에 대한 데이터 유형 호환성 (계속)

피연산자	2진 정 십진수 수	부동 소수점	문자열	그래픽 문자열	날짜	시간	시각	2진 문 자열	UDT

주:

1 날짜 시간 값과 문자열의 호환성은 지정과 비교에 제한이 있습니다.
- 날짜 시간 값은 114 페이지의 『날짜 시간 지정』에서 설명한 것처럼 문자열 컬럼과 문자열 변수에 지정될 수 있습니다.
- 날짜의 유효한 문자열 표현은 날짜 컬럼에 지정하거나 날짜와 비교할 수 있습니다.
- 시간의 유효한 문자열 표현은 시간 컬럼에 지정하거나 시간과 비교할 수 있습니다.
- 시각의 유효한 문자열 표현은 시각 컬럼에 지정하거나 시각과 비교할 수 있습니다.

2 사용자 정의 구별 유형(UDDT) 값은 UDDT가 같게 정의된 값에만 비교됩니다. 일반적으로, 지정은 구별 유형 값과 그 소스 데이터 유형간에 지원됩니다. 사용자가 정의하는 구조화 유형은 비교할 수 없고 같은 구조화 유형의 피연산자나 해당되는 수퍼 유형 중 하나에만 지정될 수 있습니다. 117 페이지의 『사용자 정의 유형 지정』에서 자세한 정보를 참조하십시오.

3 FOR BIT DATA로 정의된 문자열은 또한 2진 문자열과 호환성이 없음을 의미한다는 것을 기억하십시오.

4 DATALINK 피연산자는 다른 DATALINK 피연산자에만 지정할 수 있습니다. 컬럼이 NO LINK CONTROL로 정의되거나 파일이 존재하는 데 파일 링크 제어하에 없는 경우, DATALINK 값은 컬럼에만 지정할 수 있습니다.

5 참조 유형의 지정 및 비교에 대한 118 페이지의 『참조 유형 지정』 및 124 페이지의 『참조 유형 비교』에서 자세한 정보를 참조하십시오.

지정 조작에 대한 기본 규칙은, 널(NULL) 값은 널(NULL) 값을 포함할 수 없는 컬럼이나 연관되는 표시기 변수가 없는 호스트 변수에 지정할 수 없는 것입니다. (표시기 변수에 대한 156 페이지의 『호스트 변수 참조』에서 설명을 참조하십시오.)

숫자 지정

숫자 지정의 기본적인 규칙은 십진수 또는 정수 전체를 절단하지는 않는다는 것입니다. 목표 숫자의 스케일이 지정된 숫자의 스케일보다 작은 경우 십진수의 소수점 이하 자리에서 초과하는 자리수는 절단됩니다.

십진수나 정수에서 부동 소수점 수로

부동 소수점 수는 실수와 거의 유사합니다. 그러므로 십진수나 정수가 부동 소수점 수 컬럼이나 변수에 지정되면, 결과는 원래의 숫자와 같지 않을 수도 있습니다.

부동 소수점 수나 십진수에서 정수로

부동 소수점 수나 십진수를 정수 컬럼이나 변수로 지정하면, 숫자의 분수 부분이 없어집니다.

십진수에서 십진수로

십진수가 십진수 컬럼이나 변수로 지정되면, 필요한 경우 목표의 정밀도와 스케일로 변형됩니다. 앞에 필요한 수만큼 0이 붙거나 삭제되고, 숫자의 분수 부분에서는 뒤에 필요한 만큼 0이 붙거나 삭제됩니다.

정수에서 십진수로

정수가 십진수 컬럼이나 변수에 지정되면, 숫자는 먼저 임시 십진수로 변환된 다음, 필요하면 목표의 정밀도와 스케일로 변환됩니다. 임시 십진수의 정밀도와 스케일은 작은 정수의 경우에는 5,0이고 큰 정수의 경우에는 11,0이며 대 정수의 경우에는 19,0입니다.

부동 소수점 수에서 십진수로

부동 소수점 수가 십진수로 변환되면, 숫자는 먼저 정밀도 31로의 암시 십진수로 변환된 다음, 필요하면 목표의 정밀도와 스케일로 절단됩니다. 변환에서, 숫자는 31 십진 디지트의 정밀도로 반올림(소수 연산사용)됩니다. 결과적으로, $0.5*10^{-31}$보다 작은 숫자는 0으로 감소됩니다. 스케일은 숫자의 정수 부분이 유의값 유실 없이 표시될 수 있는 가장 큰 가능한 값을 제공합니다.

문자열 지정

두 가지 유형의 지정이 있습니다.

- *저장영역 지정(storage assignment)*은 값을 컬럼이나 함수의 매개변수에 지정할 때입니다.
- *검색 지정(retrieval assignment)*은 값을 호스트 변수에 지정할 때입니다.

문자열 지정 규칙은 지정 유형에 따라 다릅니다.

저장영역 지정(storage assignment)

기본 규칙은 컬럼이나 함수 매개변수에 지정된 문자열의 길이는 컬럼이나 함수 매개변수의 길이 속성보다 클 수 없다는 것입니다. 문자열의 길이가 컬럼이나 함수 매개변수의 길이 속성보다 큰 경우, 다음 조치가 발생합니다.

- 문자열이 절단된 후미 공백으로 지정되어(long 문자열을 제외한 모든 문자열 유형에서) 목표 컬럼 또는 함수 매개변수의 길이 속성에 맞게 됩니다.
- 다음과 같은 경우에 오류가 리턴됩니다(SQLSTATE 22001).
  - long 문자열 이외에서 비 공백 문자가 절단되는 경우
  - long 문자열에서 임의의 문자(또는 바이트)가 절단되는 경우

문자열이 고정 길이 컬럼에 지정되고, 문자열의 길이가 목표의 길이 속성보다 짧은 경우, 문자열의 오른쪽에는 필요한 만큼의 공백이 첨부됩니다. 첨부된 문자는 FOR BIT DATA 속성으로 정의된 컬럼의 경우에도 공백이 됩니다.

검색 지정(retrieval assignment)

호스트 변수에 지정된 문자열 길이가 호스트 변수의 길이 속성보다 길 수 있습니다. 문자열이 호스트 변수에 지정되고, 문자열의 길이가 변수의 길이 속성보다 긴 경우, 문자열의 오른쪽에서 필요한 문자수(또는 바이트수) 만큼 절단됩니다. 이 경우, 경고(SQLSTATE 01004)가 발생하며 'W'값이 SQLCA의 SQLWARN1 필드에 지정됩니다.

덧붙여, 표시기 변수가 제공되고 값의 소스가 LOB가 아닌 경우, 표시기 변수가 문자열의 원래 길이로 설정됩니다.

문자열이 고정 길이 변수로 지정되고 문자열의 길이가 목표의 길이 속성보다 짧은 경우, 문자열의 오른쪽에 필요한 수만큼 공백으로 채워집니다. 채운 문자는 FOR BIT DATA 속성으로 정의된 문자열의 경우에도 공백이 됩니다.

C NUL 종료 호스트 변수의 검색 지정은 PREP 또는 BIND 명령에서 지정된 옵션을 기반으로 하여 처리됩니다. 세부사항은 *응용프로그램 개발 안내서*에서 C 및 C++로 된 프로그래밍 부분을 참조하십시오.

문자열 지정을 위한 변환 규칙

컬럼 또는 호스트 변수에 지정된 문자열 또는 그래픽 문자열은 필요한 경우, 우선 목표의 코드 페이지로 변환됩니다. 문자 변환은 다음 사항이 참인 경우에만 필요합니다.

- 코드 페이지가 다릅니다.

- 문자열이 널(NULL)도 아니고 비어있지도 않습니다.
- 문자열이 0(FOR BIT DATA)의 코드 페이지값은 갖지 않습니다. [16]

문자열 지정에 대한 MBCS 고려사항

1바이트 및 복수 바이트 문자를 모두 포함할 수 있는 문자열을 지정할 경우에는 여러 가지 사항을 고려해야 합니다. 이러한 고려사항은 FOR BIT DATA로 지정된 문자열을 포함하여 모든 문자열에 적용됩니다.

- 공백 채움은 항상 1바이트 공백 문자(X'20')를 이용하여 이루어집니다.
- 공백 절단은 항상 1바이트 공백 문자(X'20')를 기반으로 하여 이루어집니다. 2바이트 공백 문자는 절단이 되면 다른 문자로 처리됩니다.
- 호스트 변수에 문자열을 지정하는 것은 목표 호스트 변수가 전체 소스 문자열을 포함하기에 크기가 충분하지 않은 경우, MBCS 문자가 세분화될 수 있습니다. MBCS 문자가 세분화되면, 목표의 MBCS 문자 단편 각각의 바이트가 1바이트 공백 문자(X'20')로 설정되고, 더 이상 소스에서 바이트가 이동되지 않으며, SQLWARN1은 'W'로 설정되어 절단을 표시합니다. 문자열이 FOR BIT DATA로 정의될 경우에도, 동일한 MBCS 문자 단편 처리가 적용된다는 점에 유의하십시오.

그래픽 문자열 지정에 대한 DBCS 고려사항

그래픽 문자열 지정은 문자열에 대한 방법과 같은 방법으로 수행됩니다. 그래픽 문자열 데이터 유형은 다른 그래픽 문자열 데이터 유형과만 호환되며, 숫자, 문자열 또는 날짜시간 데이터 유형과는 호환되지 않습니다.

그래픽 문자열 값이 그래픽 문자열 컬럼에 지정되면, 값의 길이는 컬럼의 길이보다 클 수 없습니다.

그래픽 문자열 값('소스' 문자열)이 고정 길이 그래픽 문자열 데이터 유형(컬럼 또는 호스트 변수가 될 수 있는 '목표')에 지정되고 소스 문자열의 길이가 목표 길

16. 입력 호스트 변수가 DRDA 응용프로그램 서버(AS)로 작용할 경우, 입력 호스트 변수는 지정중이거나 FOR BIT DATA 컬럼과 비교내지 결합되는 중이라도 서버의 코드 페이지로 변환됩니다. SQLDA가 수정되어 입력 호스트 변수를 FOR BIT DATA로 식별되도록 한 경우에는 변환이 수행되지 않습니다.

이보다 짧은 경우, 목표에는 소스 문자열의 사본이 포함되며, 이 문자열은 목표 길이와 같은 길이의 값을 작성하는 데 필요한 2바이트 공백 문자 수로 오른쪽에서 채워집니다.

그래픽 문자열 값이 그래픽 문자열 호스트 변수에 지정되고, 소스 문자열의 길이가 호스트 변수의 길이보다 큰 경우, 호스트 변수에는 호스트 변수의 길이와 같도록 필요한 만큼의 2바이트 문자가 절단된 소스 문자열의 사본이 들어 있습니다(이 방법의 경우, 2바이트 문자의 중간에서 절단하는 것은 고려할 필요가 없습니다. 절단이 발생하는 경우, 원시값이나 목표 호스트 변수는 모두 잘못 정의된 그래픽 문자열 데이터 유형이 됨을 기억하십시오.). SQLCA에서 경고 플래그 SQLWARN1가 'W'로 설정됩니다. 표시기 변수가 정의되었다면 소스 문자열의 원래 길이(2바이트 문자)가 들어 있습니다. 그러나 DBCLOB의 경우, 표시기 변수에는 원래 길이가 없습니다.

C NULL 종료 호스트 변수(wchart_t를 이용하여 선언된 변수)의 검색 지정은 PREP 또는 BIND 명령에서 지정된 옵션을 기반으로 하여 처리됩니다. 세부사항은 *응용 프로그램 개발 안내서*에서 C 및 C++로 된 프로그래밍 부분을 참조하십시오.

날짜 시간 지정

날짜 시간 지정에 대한 기본 규칙은, DATE, TIME 또는 TIMESTAMP 값은 일치하는 데이터 유형(DATE, TIME 또는 TIMESTAMP)이 있는 컬럼이나, 고정 또는 가변 길이 문자열 변수나 문자열 컬럼에만 지정될 수 있다는 것입니다. 지정은 LONG VARCHAR, BLOB 또는 CLOB 변수나 컬럼에는 할 수 없습니다.

날짜 시간 값이 문자열 변수나 문자열 컬럼에 지정되면, 문자열로의 변환이 자동으로 수행됩니다. 앞에 오는 0은 일, 시 또는 시각 부분에서 생략되지 않습니다. 목표의 필요 길이는 문자열 표현의 형식에 따라 다릅니다. 목표 길이가 필요한 길이보다 크고, 목표가 고정 길이 문자열인 경우, 오른쪽에 공백이 채워집니다. 목표의 길이가 필요한 길이보다 적은 경우, 결과는 포함된 날짜 시간 값의 유형과 목표의 유형에 따라 다릅니다.

목표가 호스트 변수이면, 다음 규칙이 적용됩니다.

- **DATE의 경우:** 변수 길이가 10 바이트 미만이면, 오류가 발생합니다.

- **TIME의 경우:** USA 형식이 사용될 경우, 변수의 길이는 8이상이어야 합니다. 다른 형식에서는 길이가 5 이상이여야 합니다.

  ISO나 JIS 형식을 사용하는 경우, 호스트 변수의 길이가 8 미만인 경우, 시간의 두 번째 부분은 결과에서 생략되어 표시기 변수로 지정됩니다. SQLCA의 SQLWARN1 필드는 생략을 표시하도록 설정됩니다.

- **TIMESTAMP의 경우:** 호스트 변수가 19 바이트 미만이면, 오류가 발생합니다. 길이가 26 미만이지만 19바이트 이상인 경우, 값의 마이크로초 부분의 뒤에 오는 디지트가 생략됩니다. SQLCA의 SQLWARN1 필드는 생략을 표시하도록 설정됩니다.

날짜 시간 값에 대한 문자열 길이에 대해서는 95 페이지의 『날짜 시간 값』에서 참조하십시오.

DATALINK 지정

값의 연결 속성이 비어 있거나 컬럼이 NO LINK CONTROL로 정의되어 있지 않는 한, DATALINK 컬럼에 값을 지정하면 파일에 대한 연결이 설정됩니다. 연결된 값이 이미 컬럼에 존재하는 경우 그 파일의 연결은 해제됩니다. 연결된 값이 이미 존재하는 경우 널(NULL) 값을 지정해도, 기존값과 연결된 파일의 연결이 해제됩니다.

적용업무가 컬럼의 기존 데이터 위치와 동일한 데이터 위치를 제공할 경우, 연결은 그대로 유지됩니다. 원인은 다음과 같습니다.

- 주석이 변경되는 경우
- 테이블이 DRNP(Datalink Reconcile Not Possible) 상태에 있을 경우, 컬럼에 있는 속성과 동일한 링크 속성을 제공하여 테이블의 링크를 복원할 수 있습니다.

다음과 같은 방식으로 DATALINK 값이 컬럼에 지정될 수 있습니다.

- DLVALUE 스칼라 함수를 사용하여 새로운 DATALINK 값을 작성하고 이를 컬럼에 지정할 수 있습니다. 이 값에 주석만 포함되어 있거나 URL이 완벽하게 동일하지 않는 한, 지정을 하면 파일이 연결됩니다.

- DATALINK 값은 CLI 함수 SQLBuildDataLink를 사용하여 CLI 매개변수 내에서 구성될 수 있습니다. 그런 후 이 값이 컬럼에 지정될 수 있습니다. 이 값에 주석만 포함되어 있거나 URL이 완벽하게 동일하지 않는 한, 지정을 하면 파일이 연결됩니다.

DATALINK 컬럼에 값을 지정할 때, 다음과 같은 오류 조건은 SQLSTATE 428D1을 리턴합니다.

- 데이터 위치(URL) 형식이 유효하지 않습니다(이유 코드 21).
- 파일 서버가 이 데이터베이스로 등록된 상태가 아닌 경우(이유 코드 22)
- 유효하지 않은 링크 유형이 지정된 경우(이유 코드 23)
- 주석 또는 URL 길이가 유효하지 않은 경우(이유 코드 27)

  URL 매개변수 또는 함수 결과의 크기는 입출력시 동일하며 DATALINK 컬럼의 길이에 의해 바인드됩니다. 그러나, 경우에 따라서는 리턴되는 URL 값에 액세스 토큰이 접속되어 있는 경우가 있습니다. 이것이 가능한 상황에서는 출력 위치에 액세스 토큰 및 DATALINK 컬럼의 길이를 수용할 만한 저장영역 공간이 있어야 합니다. 따라서, 기본 URL 스키마 또는 기본 호스트 이름을 포함하여 입력에 제공된 완전히 확장된 주석 및 URL의 실제 길이는 출력 저장영역 공간에 들어갈 수 있도록 제한되어야 합니다. 제한 길이가 초과되면, 이 오류가 발생합니다.

지정 결과 링크도 작성될 때 다음과 같은 오류가 발생할 수 있습니다.

- 파일 서버가 현재 사용 가능하지 않습니다(SQLSTATE 57050).
- 파일이 존재하지 않습니다(SQLSTATE 428D1, 이유 코드 24).
- 참조된 파일을 링크하기 위해 액세스할 수 없습니다(이유 코드 26).
- 파일이 이미 다른 컬럼에 링크됩니다(SQLSTATE 428D1, 이유 코드 25)

  다른 데이터베이스에 대한 링크인 경우에도 이러한 오류가 발생할 수 있습니다.

또한, 지정 결과 기존의 링크가 제거되는 경우 다음과 같은 오류가 발생할 수 있습니다.

- 파일 서버가 현재 사용 가능하지 않습니다(SQLSTATE 57050).
- 데이터 링크 파일 관리 프로그램에 따라, 참조 무결성 제어를 가진 파일이 올바른 상태가 아닙니다(SQLSTATE 58004).

다음과 같은 방식으로 DATALINK 값을 데이터베이스로부터 검색할 수 있습니다.

- 스칼라 함수(예. DLLINKTYPE 또는 DLURLPATH)를 사용하여 DATALINK 값의 일부를 호스트 변수에 지정할 수 있습니다.

일반적으로, 검색할 때 파일 서버에 액세스하려는 시도가 이루어지지 않습니다. [17] 따라서, 파일 시스템 명령을 통해 파일 서버에 액세스하려는 후속 시도는 실패할 수도 있습니다.

DATALINK 검색시, 파일 서버가 아직 그 데이터베이스 서버에 등록되어 있는지를 확인하기 위해 데이터베이스 서버에 있는 파일 서버의 등록을 점검합니다 (SQLSTATE 55022). 또한, 테이블이 조정 보류 상태이거나 조정이 가능하지 않은 상태이므로, DATALINK 값을 검색할 때 경고를 표시할 수 있습니다 (SQLSTATE 01627).

사용자 정의 유형 지정

사용자 정의 유형의 경우, 다른 모든 지정에 사용되는 것과는 다른 규칙이 호스트 변수에 대한 지정에 적용됩니다.

구별 유형: 호스트 변수에 대한 지정은 구별 유형의 소스 유형에 따라 수행됩니다. 즉, 다음 규칙을 준수합니다.

- 지정의 오른쪽에 있는 구별 유형값은 이 구별 유형의 소스 유형을 이 호스트 변수에 지정할 수 있는 경우 왼쪽에 있는 호스트 변수로 지정할 수 있습니다.

지정의 목표가 구별 유형을 기초로 하는 컬럼일 경우, 소스 데이터 유형은 사용자 정의 유형에 대해 106 페이지의 『데이터 유형간의 변환』에 설명된 대로 목표 데이터로 유형변환할 수 있어야 합니다.

구조화 유형: 호스트 변수에서, 그리고 호스트 변수로의 지정은 호스트 변수의 선언된 유형을 기초로 합니다. 즉, 다음 규칙을 준수합니다.

---

17. 경로와 연결된 접두부를 결정하기 위해 파일 서버에 액세스해야 할 수도 있습니다. 이는 파일 시스템의 마운트 위치가 이동될 때 파일 서버에서 변경될 수 있습니다. 서버상에서 파일에 처음 액세스하면 필요한 값들이 파일 서버로부터 검색되고 데이터베이스 서버에서 캐쉬되어, 그 파일 서버에 대한 이후의 DATALINK 값 검색이 이루어지게 됩니다. 파일 서버를 액세스할 수 없는 경우 오류가 리턴됩니다(SQLSTATE 57050).

지정 오른쪽에 있는 구조화 유형 값은 호스트 변수의 선언된 유형이 구조화 유형이거나 구조화 유형의 수퍼 유형인 경우에 왼쪽에 있는 호스트 변수에 지정할 수 있습니다.

지정의 목표가 구조화 유형 컬럼일 경우, 소스 데이터 유형은 목표 데이터 유형이나 목표 데이터 유형의 수퍼 유형이어야 합니다.

참조 유형 지정

목표 유형이 *T*인 참조 유형은 목표 유형이 *S*인 참조 유형이기도 한 참조 유형 컬럼에 지정될 수 있습니다. 여기서 *S*는 *T*의 수퍼 유형입니다. 범위 지정된 참조 컬럼이나 변수에 지정될 경우, 지정되는 실제 값이 범위에 의해 정의된 목표 테이블이나 뷰에 존재하는지 확인하기 위한 점검이 수행되지 않습니다.

호스트 변수에 대한 지정은 참조 유형의 표시 유형에 따라 수행됩니다. 즉, 다음 규칙을 준수합니다.

- 지정 오른쪽에 있는 참조 유형 값은 이 참조 유형의 표시 유형이 이 호스트 변수에 지정 가능한 경우, 왼쪽에 있는 호스트 변수에 지정할 수 있습니다.

지정 목표가 컬럼이고 지정 오른쪽에 호스트 변수가 있는 경우, 그 호스트 변수는 목표 컬럼의 참조 유형으로 명시적으로 유형변환(cast)되어야 합니다.

숫자 비교

숫자는 대수학적, 즉, 부호에 따라 비교됩니다. 예를 들어, -2는 +1 보다 작습니다.

한 숫자가 정수이고, 나머지는 십진수인 경우, 십진수로 변환된 정수의 임시 사본으로 비교합니다.

스케일이 다른 십진수를 비교할 때, 숫자 중 하나에서 분수 부분이 다른 숫자와 같은 디지트가 되도록 뒤에 0을 추가한 임시 복사본과 비교합니다.

한 숫자가 부동 소수점 수이고, 나머지는 정수거나 십진수인 경우, 다른 유형의 숫자의 임시 사본으로 비교합니다. 이때 사본은 배밀도 부동 소수점으로 변환된 것입니다.

두 개의 부동 소수점 수는 이들의 정규화된 양식의 비트 구성이 동일할 때에만 같습니다.

문자열 비교

항상 비트 값으로 비교되는 FOR BIT DATA 속성을 갖는 문자열을 제외하고는 문자열은 항상 데이터베이스가 작성될 때 지정된 조합 순서에 따라 비교됩니다.

길이가 틀린 문자열을 비교할 때, short 문자열을 long 문자열의 길이에 맞도록 왼쪽에 1바이트 공백을 채운 논리 사본을 이용하여 비교합니다. 논리 확장은 FOR BIT DATA로 태그가 붙은 문자열을 포함한 모든 문자열에 대해 이루어집니다.

문자열(FOR BIT DATA로 태그가 붙어 있는 문자열은 제외)은 데이터베이스 작성시 지정되는 조합 순서에 따라 비교됩니다(데이터베이스 작성시 지정되는 조합 순서에 대한 *관리 안내서*에서 좀더 자세한 정보를 참조하십시오.). 예를 들어, 데이터베이스 관리 프로그램에 의해 제공되는 기본 조합 순서는 같은 문자, 같은 가중치의 소문자 및 대문자 버전을 제공할 수도 있습니다. 데이터베이스 관리 프로그램은 두 단계 비교를 수행하여 하나의 동일 문자열만 같은 것으로 간주되도록 합니다. 첫번째 단계에서, 문자열은 데이터베이스 조합 순서에 따라 비교됩니다. 문자열에 있는 문자의 비중이 같은 경우, 두 번째 "타이-브레이커(tie-breaker)" 통과가 수행되어 이들의 실제 코드값을 기준으로 문자열을 비교합니다.

두 문자열 모두가 비어 있거나 해당 바이트가 같으면 두 문자열은 같습니다. 두 피연산자 중 하나가 널(NULL)인 경우, 결과는 알 수 없습니다.

long 문자열 및 LOB 문자열은 기본 비교 연산자(=, <>, <, >, <=, >=)를 사용하는 비교 연산에서는 지원되지 않습니다. LIKE 술어와 POSSTR 함수를 사용하는 비교에서는 지원됩니다. 227 페이지의 『LIKE 술어』 및 372 페이지의 『POSSTR』에서 자세한 설명을 참조하십시오.

최대 4 000바이트까지의 long 문자열과 LOB 문자열 부분은 SUBSTR 및 VARCHAR 스칼라 함수를 사용하여 비교할 수 있습니다. 예를 들어, 다음 컬럼이 제공되면,

```
MY_SHORT_CLOB   CLOB(300)
MY_LONG_VAR     LONG VARCHAR
```

다음은 유효합니다.

```
WHERE VARCHAR(MY_SHORT_CLOB) > VARCHAR(SUBSTR(MY_LONG_VAR,1,300))
```

예:

이들 예의 경우, 'A', 'Á', 'a' 및 'á'가 각각 코드 포인트 값 X'41', X'C1', X'61' 및 X'E1'을 가집니다.

문자 'A', 'Á', 'a', 'á'의 비중이 136, 139, 135 및 138인 조합 순서를 생각해 보십시오. 문자는 다음과 같이 가중치순으로 정렬됩니다.

```
'a' < 'A' < 'á' < 'Á'
```

이제 각각 0xC141, 0xC161, 0xE141 및 0xE161의 코드 포인트를 가지는 네 개의 DBCS 문자 D1, D2, D3 및 D4를 고려하십시오. 이들 DBCS 문자가 CHAR 컬럼에 있으면, 그러한 바이트의 배열 비중에 따라 바이트의 순차로 정렬됩니다. 첫 번째 바이트는 138과 139의 비중을 가지며, 따라서 D3과 D4가 D2와 D1 앞에 옵니다. 두 번째 바이트는 135와 136의 비중을 가집니다. 그러므로, 순서는 다음과 같습니다.

```
D4 < D3 < D2 < D1
```

그러나, 비교되는 값의 속성이 FOR BIT DATA일 경우나, 이 DBCS 문자들이 GRAPHIC 컬럼에 저장된 경우, 조합 가중치는 무시되고, 문자들은 다음과 같이 코드 포인트에 따라 비교됩니다.

```
'A' < 'a' < 'Á' < 'á'
```

DBCS 문자들은 다음과 같이 코드 포인트순으로, 바이트 순서로서 정렬됩니다.

```
D1 < D2 < D3 < D4
```

문자 'A', 'Á', 'a', 'á'의 비중이 74, 75, 74 및 75(고유하지 않음)인 조합 순서를 생각해 보십시오. 배열 비중만을 고려하면(첫번째 패스), 'a'는 'A'와 같고, 'á'는 'Á'와 같습니다. 문자들의 코드 포인트는 다음과 같이 타이 브레이크(tie break), (두 번째 통과)에 사용됩니다.

```
'A' < 'a' < 'Á' < 'á'
```

CHAR 컬럼에 있는 DBCS 문자는 비중에 따라(첫번째 패스), 그런 후 매듭을 풀기 위한(두 번째 패스) 코드 포인트에 따라 바이트의 순차를 정렬합니다. 첫번째 바이트는 동일한 비중을 가지므로, 코드 포인트(0xC1 및 0xE1)가 매듭을 풉니다. 그러므로, 문자 D1 및 D2가 문자 D3 및 D4에 앞서 정렬됩니다. 두 번째 바이트도 유사한 방법으로 비교되며, 결과는 다음과 같습니다.

```
D1 < D2 < D3 < D4
```

마찬가지로, CHAR 컬럼의 데이터 속성이 FOR BIT DATA일 경우나, DBCS 문자들이 GRAPHIC 컬럼에 저장된 경우, 조합 가중치는 무시되고, 문자들은 다음과 같이 코드 포인트에 따라 비교됩니다.

```
D1 < D2 < D3 < D4
```

이 특정 예의 경우, 결과가 배열 비중이 사용되었을 때와 동일해 질 수 있습니다. 이전에는 항상 그렇지는 않았습니다.

비교를 위한 변환 규칙

두 문자열을 비교할 때, 필요하다면 한 문자열이 다른 문자열의 코드 페이지 집합으로 변환됩니다. 129 페이지의 『문자열 변환 규칙』에서 자세한 설명을 참조하십시오.

결과의 순서화

정렬해야 하는 결과는 119 페이지의 『문자열 비교』에 설명된 문자열 비교 규칙을 기초로 정렬됩니다. 비교는 데이터베이스 서버에서 수행됩니다. 클라이언트 응용프로그램 결과를 리턴할 때, 코드 페이지 변환이 수행될 수 있습니다. 이 코드 페이지 변환은 서버가 결정한 결과 집합의 순서에 영향을 미치지 않습니다.

문자열 비교에 대한 MBCS 고려사항

혼합 SBCS/MBCS 문자열은 데이터베이스 작성시 지정된 조합 순서에 따라 비교됩니다. 기본(SYSTEM) 조합 순서에 의해 작성된 데이터베이스의 경우, 모든 1바이트의 ASCII 문자는 올바른 순서로 정렬되나 2바이트 문자는 반드시 코드 포인트 순서로 정렬되지는 않습니다. IDENTITY 순서를 사용하여 작성된 데이터베이스의 경우, 모든 2바이트 문자 세트도 그 코드 포인트 순서대로 제대로 정렬되지만 1바이트 ASCII 문자도 그 코드 포인트 순서대로 정렬됩니다. COMPATIBILITY 순서에 의해 작성된 데이터베이스의 경우, 대부분의 2바이트

문자를 제대로 정렬하는 데는 절충(compromise) 순서가 사용되며 ASCII에 대해서도 거의 정확합니다. 이것이 DB2 버전 2의 기본 배열 테이블입니다.

혼합 문자열은 바이트별로 비교됩니다. 각 바이트를 독립적으로 간주하므로, 혼합 문자열에서 발생하는 복수 바이트 문자의 경우 예상치 못한 결과가 발생할 수 있습니다.

예:

이 예에서 'A', 'B', 'a' 및 'b' 2바이트 문자에는 코드값 X'8260', X'8261', X'8281' 및 X'8282'가 각각 들어 있습니다.

코드 포인트 X'8260', X'8261', X'8281', X'8282'의 가중치가 96, 65, 193, 194인 조합 순서를 고려해 보십시오. 그러면,

```
'B' < 'A' < 'a' < 'b'
```

와

```
'AB' < 'AA' < 'Aa' < 'Ab' < 'aB' < 'aA' < 'aa' < 'ab'
```

그래픽 문자열 비교는 문자열에 대한 방법과 같은 방법으로 수행됩니다.

그래픽 문자열 비교는 LONG VARGRAPHIC을 제외한 모든 그래픽 문자열 데이터 유형에서 유효합니다. LONG VARGRAPHIC과 DBCLOB 데이터 유형은 비교 조작에서는 허용되지 않습니다.

그래픽 문자열의 경우, 데이터베이스의 조합 순서는 사용되지 않습니다. 대신, 그래픽 문자열은 언제나 해당 바이트의 숫자(2진)값을 기준으로 비교합니다.

이전 예를 사용할 경우, 리터럴이 그래픽 문자열이면 다음과 같습니다.

```
'A' < 'B' < 'a' < 'b'
```

와

```
'AA' < 'AB' < 'Aa' < 'Ab' < 'aA' < 'aB' < 'aa' < 'ab'
```

길이가 같지 않은 그래픽 문자열을 비교할 때, long 문자열에 맞추기 위해 오른쪽을 2바이트 공백 문자로 채운 short 문자열의 논리적 사본을 사용하여 비교를 수행합니다.

두 개의 그래픽 값은 모두가 비었거나 모든 해당 그래픽이 같을 경우 같습니다. 두 피연산자 중 하나가 널(NULL)인 경우, 결과는 알 수 없습니다. 두 값이 같지 않은 경우, 이들의 관계는 간단한 2진 문자열 비교로 결정됩니다.

이 절에 표시된 대로, 바이트 기준으로 문자열을 비교하는 것은 예상치 않은 결과를 가져올 수 있습니다. 즉, 문자 기준 비교시 예상되는 것과 다른 결과가 가능합니다. 여기에 나와 있는 예에서는 동일한 MBCS 코드 페이지를 가정하지만, 상황은 동일한 언어를 갖는 다른 복수 바이트 코드 페이지를 사용할 때 더 복잡해질 수 있습니다. 예를 들면, 일본어 DBCS 코드 페이지와 일본어 EUC 코드 페이지의 문자열을 비교하는 경우를 고려해 보십시오.

날짜 시간 비교

DATE, TIME 또는 TIMESTAMP 값은 같은 데이터 유형의 다른 값이나 같은 데이터 유형의 문자열 표현과 비교됩니다. 모든 비교는 시간순으로 진행되며, 이것은 1월 1, 0001에서 멀수록 시간의 값이 커집니다.

TIME 값과 시간값의 문자열 표현의 비교에는 언제나 초가 포함됩니다. 문자열 표현에서 초가 누락된 경우, 0초를 의미합니다.

TIMESTAMP 값이 포함된 비교는 그 표현이 같은 것으로 간주될 수 있느냐에 관계없이 연대순입니다.

예:

```
TIMESTAMP('1990-02-23-00.00.00') > '1990-02-22-24.00.00'
```

사용자 정의 유형 비교

사용자 정의 구별 유형의 값은 정확하게 동일한 사용자 정의 구별 유형의 값에만 비교될 수 있습니다. 사용자 정의 구별 유형은 WITH COMPARISONS 절을 사용하여 정의되어 있어야 합니다.

예:

다음과 같은 YOUTH 구별 유형과 CAMP_DB2_ROSTER 테이블이 제공될 경우,

```
CREATE DISTINCT TYPE YOUTH AS INTEGER WITH COMPARISONS
CREATE TABLE CAMP_DB2_ROSTER
   ( NAME                VARCHAR(20),
     ATTENDEE_NUMBER     INTEGER NOT NULL,
     AGE                 YOUTH,
     HIGH_SCHOOL_LEVEL   YOUTH)
```

다음 비교는 유효합니다.

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > HIGH_SCHOOL_LEVEL
```

다음 비교는 유효하지 않습니다.

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > ATTENDEE_NUMBER
```

그러나 AGE는 구별 유형과 소스 유형 사이에서 변환 함수 또는 CAST 스펙을 사용하여 ATTENDEE_NUMBER과 비교할 수 있습니다. 다음 비교는 모두 유효합니다.

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE INTEGER(AGE) > ATTENDEE_NUMBER
SELECT * FROM CAMP_DB2_ROSTER
   WHERE CAST( AGE AS INTEGER) > ATTENDEE_NUMBER
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > YOUTH(ATTENDEE_NUMBER)
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > CAST(ATTENDEE_NUMBER AS YOUTH)
```

사용자가 정의하는 구조화 유형의 값은 다른 값과 비교할 수 없습니다. NULL 술어와 TYPE 술어은 사용할 수 있습니다.

참조 유형 비교

참조 유형 값은 목표 유형에 공통 상위 유형(supertype)이 있는 경우에만 비교가 가능합니다. 공통 상위 유형의 스키마 이름이 함수 경로에 포함되어 있는 경우에만 적절한 비교 함수를 찾을 수 있습니다. 비교는 참조 유형의 표시 유형을 사용하여 수행합니다. 비교에서 참조 영역은 고려되지 않습니다.

결과 데이터 유형 규칙

결과의 데이터 유형은 연산에서 피연산자에 적용되는 규칙에 의해 결정됩니다. 이 절에서는 다음 규칙을 설명합니다.

이러한 규칙이 적용됩니다.

- 집합 연산자의 fullselect에 있는 해당 컬럼(UNION, INTERSECT 및 EXCEPT)
- CASE 표현식의 결과 표현식
- 스칼라 함수 COALESCE(또는 VALUE)의 인수
- IN 술어 목록에 있는 표현식 값
- 복수 행 VALUES절의 해당 표현식

이러한 규칙들은 다양한 연산에 대해 long 문자열과 LOB 문자열에 대한 기타 제한사항에도 적용됩니다.

다양한 데이터 유형이 포함되는 규칙은 다음과 같습니다. 이런 경우, 테이블이 가능한 결과 데이터 유형을 표시하는 데 사용됩니다.

이러한 테이블은 적용 가능한 길이 또는 정밀도 및 스케일을 포함하여 결과의 데이터 유형을 나타냅니다. 결과 유형은 피연산자에 따라 결정됩니다. 한쌍 이상의 피연산자가 있는 경우, 처음 피연산자부터 시작하십시오. 이것으로 다음 피연산자에 대해 고려하고, 다음 결과 유형을 결정하는 결과 유형을 얻게 됩니다. 마지막 중간 결과 유형과 마지막 피연산자가 연산의 결과 유형을 결정합니다. 연산의 처리는 왼쪽에서 오른쪽으로 수행되므로 연산이 반복될 때 중간 결과 유형이 매우 중요합니다. 예를 들면, 다음과 같은 상황을 고려하십시오.

```
CHAR(2) UNION CHAR(4) UNION VARCHAR(3)
```

첫번째 쌍은 CHAR(4) 유형으로 결과가 나옵니다. 결과 값은 항상 4개의 문자를 갖습니다. 최종 결과 유형은 VARCHAR(4)입니다. 첫번째 UNION 연산의 결과 값의 길이는 항상 4가 될 것입니다.

문자열

문자열은 다음 문자열과 호환됩니다. 문자열에는 데이터 유형 CHAR, VARCHAR, LONG VARCHAR 및 CLOB가 포함됩니다.

하나의 피연산자가 다음이면...	다른 피연산자는 다음과 같습니다...	결과의 데이터 유형은 ...입니다.
CHAR(x)	CHAR(y)	CHAR(z) 여기서, z = max(x,y)
CHAR(x)	VARCHAR(y)	VARCHAR(z) where z = max(x,y)
VARCHAR(x)	CHAR(y) 또는 VARCHAR(y)	VARCHAR(z) where z = max(x,y)
LONG VARCHAR	CHAR(y), VARCHAR(y) 또는 LONG VARCHAR	LONG VARCHAR
CLOB(x)	CHAR(y), VARCHAR(y) 또는 CLOB(y)	CLOB(z) where z = max(x,y)
CLOB(x)	LONG VARCHAR	CLOB(z) where z = max(x,32700)

결과 문자열의 코드 페이지는 129 페이지의 『문자열 변환 규칙』에 의해 결정됩니다.

그래픽 문자열

그래픽 문자열은 다른 그래픽 문자열과 호환됩니다. 그래픽 문자열에는 데이터 유형 GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC 및 DBCLOB가 포함됩니다.

하나의 피연산자가 다음이면...	다른 피연산자는 다음과 같습니다...	결과의 데이터 유형은 ...입니다.
GRAPHIC(x)	GRAPHIC(y)	GRAPHIC(z) 여기서, z = max(x,y)
VARGRAPHIC(x)	GRAPHIC(y) 또는 VARGRAPHIC(y)	VARGRAPHIC(z) where z = max(x,y)
LONG VARGRAPHIC	GRAPHIC(y), VARGRAPHIC(y) 또는 LONG VARGRAPHIC	LONG VARGRAPHIC
DBCLOB(x)	GRAPHIC(y), VARGRAPHIC(y) 또는 DBCLOB(y)	DBCLOB(z) where z = max (x,y)
DBCLOB(x)	LONG VARGRAPHIC	DBCLOB(z) where z = max (x,16350)

결과 그래픽 문자열의 코드 페이지는 129 페이지의 『문자열 변환 규칙』에 의해 결정됩니다.

2진 대형 오브젝트(BLOB)

BLOB는 다른 BLOB와만 호환되며, 결과도 BLOB입니다. BLOB 유형으로 취급해야 할 경우, 다른 유형을 변환할 때, BLOB 스칼라 함수를 사용해야 합니다(289 페이지의『BLOB』참조). 결과 BLOB의 길이는 모든 데이터 유형의 최대 길이입니다.

숫자

숫자 유형은 다른 숫자 유형과 호환됩니다. 숫자 유형으로는 SMALLINT, INTEGER, BIGINT, DECIMAL, REAL 및 DOUBLE이 있습니다.

하나의 피연산자가 다음이면...	다른 피연산자는 다음과 같습니다...	결과의 데이터 유형은 ...입니다.
SMALLINT	SMALLINT	SMALLINT
INTEGER	INTEGER	INTEGER
INTEGER	SMALLINT	INTEGER
BIGINT	BIGINT	BIGINT
BIGINT	INTEGER	BIGINT
BIGINT	SMALLINT	BIGINT
decimal(w,x)	SMALLINT	decimal(p,x) 여기서, p = x+max(w-x,5)[1]
decimal(w,x)	INTEGER	decimal(p,x) 여기서, p = x+max(w-x,11)[1]
decimal(w,x)	BIGINT	decimal(p,x) 여기서, p = x+max(w-x,19)[1]
decimal(w,x)	decimal(y,z)	decimal(p,s) 여기서, p = max(x,z)+max(w-x,y-z)[1]s = max(x,z)
REAL	REAL	REAL
REAL	DECIMAL, BIGINT, INTEGER, or SMALLINT	DOUBLE
DOUBLE	숫자	DOUBLE

주: 1. 정밀도는 31을 초과할 수 없습니다.

DATE

날짜는 다른 날짜 또는 날짜의 유효한 표현식이 들어 있는 CHAR이나 VARCHAR 표현식과 호환됩니다. 결과의 데이터 유형은 DATE입니다.

TIME

시간(time)은 다른 시간 또는 시간의 유효한 표현식이 들어 있는 CHAR이나 VARCHAR 표현식과 호환됩니다. 결과의 데이터 유형은 TIME입니다.

TIMESTAMP

시간소인은 다른 시간소인 또는 시간소인의 유효한 표현식이 들어 있는 CHAR이나 VARCHAR 표현식과 호환됩니다. 결과의 데이터 유형은 TIMESTAMP입니다.

DATALINK

데이터 링크는 다른 데이터 링크와 호환됩니다. 결과의 데이터 유형은 DATALINK입니다. 결과 DATALINK 길이는 모든 데이터 유형의 최대 길이입니다.

사용자 정의 유형

구별 유형

사용자 정의 구별 유형은 동일한 사용자 정의 구별 유형과만 호환됩니다. 결과의 데이터 유형은 사용자 정의 구별 유형입니다.

참조 유형

목표 유형에 공통 상위 유형이 있는 경우, 참조 유형은 다른 참조 유형과 호환됩니다. 결과의 데이터 유형은 목표 유형으로서 공통 상위 유형을 가진 참조 유형입니다. 모든 피연산자에 동일한 영역 테이블이 있는 경우 결과는 그 영역 테이블을 갖게 됩니다. 그렇지 않은 경우 결과의 영역 지정은 취소(unscope)됩니다.

구조화 유형

구조화 유형에 공통 상위 유형이 있는 경우, 구조화 유형은 다른 참조 유형과 호환됩니다. 결과로 생성되는 구조화 유형 컬럼의 정적 데이터 유형은 어느 한 컬럼의 최소 공통 수퍼 유형인 구조화 유형입니다.

예를 들어, 다음과 같은 구조화 유형 계층을 고려해 보십시오.

```
A
     / \
    B   C
   / \
  D   E
 / \
F   G
```

정적 유형 E 및 F의 구조화 유형은 E 및 F의 최소 공통 수퍼 유형인, B에 대해 결과로 생성되는 정적 유형과 호환됩니다.

결과의 널(NULL) 가능 속성

INTERSECT과 EXCEPT를 제외하고, 결과에서는 두 피연산자가 널(NULL)을 허용하지 않는 경우 이외에는 널(NULL)을 허용합니다.

- INTERSECT의 경우 피연산자 중 하나가 널(NULL)을 허용하지 않는 경우, 결과에서는 널(NULL)을 사용할 수 없습니다(교집합은 널(NULL)이 될 수 없습니다.).
- EXCEPT의 경우 첫번째 피연산자가 널(NULL)을 허용하지 않는 경우, 결과에서도 널(NULL)을 허용하지 않습니다(결과는 첫번째 피연산자 값만 될 수 있습니다.).

문자열 변환 규칙

연산을 수행하는 데 사용된 코드 페이지는 이 연산에서의 피연산자에 적용된 규칙으로 결정됩니다. 이 절에서는 다음 규칙을 설명합니다.

이러한 규칙이 적용됩니다.

- 집합 연산(UNION, INTERSECT 및 EXCEPT)이 있는 fullselect의 해당 문자열 컬럼
- 병합의 피연산자
- 술어 피연산자(LIKE 제외)
- CASE 표현식의 결과 표현식
- 스칼라 함수 COALESCE(또는 VALUE)의 인수
- IN 술어 목록에 있는 표현식 값

- 복수 행 VALUES절의 해당 표현식

각각의 경우 결과의 코드 페이지는 바인드시에 결정되고, 연산 실행에는 문자열을 코드 페이지가 식별하는 코드 페이지로 변환하는 작업이 포함됩니다. 유효한 변환이 없는 문자는 문자 집합에 대한 대체 문자에 대응되며, SQLWARN10은 SQLCA의 'W'로 설정됩니다.

결과의 코드 페이지는 피연산자의 코드 페이지에 의해 결정됩니다. 첫번째 두 피연산자의 코드 페이지가 임시 결과 코드 페이지를 결정하고, 다음 피연산자의 코드 페이지와 이 코드 페이지가 새로운 임시 결과 코드 페이지를 결정합니다. 마지막 임시 결과 코드 페이지와 마지막 피연산자의 코드 페이지가 결과 문자열이나 컬럼의 코드 페이지를 결정합니다. 각 코드 페이지의 경우, 결과는 다음 규칙의 순차적인 적용에 의해 결정됩니다.

- 코드 페이지가 같은 경우, 결과는 그 코드 페이지입니다.
- 어느 한 코드 페이지가 BIT DATA(코드 페이지 0)일 경우, 결과 코드 페이지는 BIT DATA입니다.
- 이 외의 경우, 결과 코드 페이지는 표8에 의해 결정됩니다. 테이블 내의 '첫번째' 항목은 첫번째 피연산자에서 선택되는 코드 페이지를 의미하고, '두 번째' 항목은 두 번째 피연산자에서 선택되는 코드 페이지를 의미합니다.

표 8. 임시 결과의 코드 페이지 선택

첫번째 피연산자	두 번째 피연산자				
	컬럼값	추출값	상수	특수 레지스터	호스트 변수
컬럼값	첫번째	첫번째	첫번째	첫번째	첫번째
추출값	두 번째	첫번째	첫번째	첫번째	첫번째
상수	두 번째	두 번째	첫번째	첫번째	첫번째
특수 레지스터	두 번째	두 번째	첫번째	첫번째	첫번째
호스트 변수	두 번째	두 번째	두 번째	두 번째	첫번째

임시 결과는 추출 값 피연산자가 되는 것으로 간주합니다. 하나의 컬럼값, 상수, 특수 레지스터 또는 호스트 변수가 아닌 표현식도 추출값 피연산자로 간주됩니다. 표현식이 CAST 스펙(또는 같은 함수의 호출)인 경우는 여기에서 제외됩니다. 이런 경우, 첫번째 피연산자에 대한 종류는 CAST 스펙의 첫번째 인수를 기준으로 합니다.

뷰 컬럼에는 가장 기본이 되는 오브젝트의 피연산자 유형이 있는 것을 간주합니다. 예를 들어, 테이블 컬럼에서 정의된 뷰 컬럼이 컬럼값으로 간주되는 반면, 문자열 표현식(예를 들면, A CONCAT B)을 기준으로 하는 뷰 컬럼은 추출 값으로 간주됩니다.

필요하면, 다음에 대해 결과의 코드 페이지 변환이 수행됩니다.

- 병합 연산자의 피연산자
- COALESCE(또는 VALUE) 스칼라 함수의 선택된 인수
- CASE 표현식의 선택된 결과 표현식
- IN 술어 목록에 있는 표현식
- 복수 행 VALUES절의 해당 표현식
- 집합 연산에 포함된 해당 컬럼

문자 변환은 다음 사항이 참인 경우에만 필요합니다.

- 코드 페이지가 다릅니다.
- 문자열이 BIT DATA가 아닙니다.
- 문자열이 널(NULL)이 아니고 비어 있지도 않습니다.
- 코드 페이지 변환 테이블이 변환이 필요함을 나타냅니다.

예:

예 1: 다음이 제공될 경우:

표현식	유형	코드 페이지
COL_1	컬럼	850
HV_2	호스트-변수	437

술어을 평가할 경우:

```
COL_1 CONCAT :HV_2
```

두 피연산자의 결과 코드 페이지는 850입니다. 이것은 중요한 피연산자가 컬럼 COL_1이기 때문입니다.

예 2: 술어를 평가할 때, 이전 예의 정보를 사용할 경우:

```
COALESCE(COL_1, :HV_2:NULLIND,)
```

결과 코드 페이지는 850입니다. 그러므로, COALESCE 스칼라 함수의 결과 코드 페이지는 코드 페이지 850이 됩니다.

파티션 호환성

*파티션 호환성(Partition compatibility)*은 파티션 키의 해당 컬럼의 기본 데이터 유형들 사이에서 정의됩니다. 파티션 호환 데이터 유형은 한 유형에서 하나씩 두 개의 변수가 동일한 파티션 함수에 의해 동일한 파티션 맵에 대응되는 등록 정보를 갖습니다.

133 페이지의 표9는 파티션에서의 데이터 유형의 호환 내역을 보여줍니다.

파티션 호환성은 다음 특성을 갖습니다.

- 내부 형식은 DATE, TIME 및 TIMESTAMP에 사용됩니다. 이들은 서로 호환되지 않으며, 이들 중 어느 것도 CHAR과 호환되지 않습니다.
- 파티션 호환은 NOT NULL 또는 FOR BIT DATA 정의가 있는 컬럼의 영향을 받지는 않습니다.
- 호환되는 데이터 유형의 널(NULL) 값들은 동일하게 취급됩니다. 호환되지 않는 데이터 유형의 널(NULL) 값들에 대해서는 서로 다른 결과가 산출될 수 있습니다.
- UDT의 기본 데이터 유형은 파티션 호환 가능성을 분석하는 데 사용됩니다.
- 파티션 키에 있는 동일한 값의 십진수들은 이들의 스케일과 정밀도가 다르더라도 동등하게 취급됩니다.
- 문자열(CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC)에 있는 뒤 공백은 시스템이 제공하는 해쉬 기능에 의해 무시됩니다.
- 길이가 서로 다른 CHAR VARCHAR은 호환 가능한 데이터 유형입니다.
- 서로 같은 REAL이나 DOUBLE 값은 정밀도가 다르더라도 동등하게 취급됩니다.

표 9. 파티션 호환성

피연산자	2진 정수	십진수	부동 소수점	문자열	그래픽 문자열	날짜	시간	시각	구별 유형	구조화 유형
2진 정수	예	No	No	No	No	No	No	No	[1]	No
십진수	No	예	No	No	No	No	No	No	[1]	No
부동 소수점	No	No	예	No	No	No	No	No	[1]	No
문자열[3]	No	No	No	예[2]	No	No	No	No	[1]	No
그래픽 문자열[3]	No	No	No	No	예	No	No	No	[1]	No
날짜	No	No	No	No	No	예	No	No	[1]	No
시간	No	No	No	No	No	No	예	No	[1]	No
시각	No	No	No	No	No	No	No	예	[1]	No
구별 유형	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	No
구조화 유형[3]	No	No	No	No	No	No	No	No	No	No

주:

1 사용자 정의 구별 유형(UDT) 값은 UDT의 소스 유형이나, 파티션 호환 가능 소스 유형의 다른 UDT와 호환 가능한 파티션입니다.

2 FOR BIT DATA 속성은 파티션 호환성에 영향을 주지 않습니다.

3 사용자가 정의하는 구조화 유형 및 데이터 유형 LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB, BLOB는 파티션 키에서 지원되지 않으므로 파티션 호환성에 적용할 수 없습니다.

상수

상수(때때로 *리터럴(literal)*이라고도 함)는 값을 지정합니다. 상수는 문자열이나 상수로 구분됩니다. 숫자 상수는 정수, 부동 소수점 수 또는 십진수로 다시 분류됩니다.

모든 상수에는 속성 NOT NULL이 있습니다.

숫자 상수에 있는 음수 0값(-0)은 부호가 없는 제로(0)와 같은 값입니다.

정수 상수

*정수 상수*는 최대 디지트가 19(소수점은 포함되지 않음)인 부호있는 또는 부호없는 숫자로서 정수를 지정합니다. 정수 상수의 데이터 유형 값이 큰 정수 범위 내에 있으면 큰 정수입니다. 정수 상수의 데이터 유형 값이 큰 정수 범위 밖에 있으나 대 정수 범위 내에 있으면 이는 대 정수입니다. 대 정수값의 범위를 벗어나 정의된 상수는 십진 상수로 간주됩니다.

큰 정수 상수의 가장 작은 리터럴 표시는 -2 147 483 647로, 정수 값의 한계인 -2 147 483 648이 아닙니다. 마찬가지로, 대 정수 상수의 최소 리터럴 표현은 -9 223 372 036 854 775 807이지 대 정수값에 대한 한계인 -9 223 372 036 854 775 808이 아닙니다.

예

```
64      -15      +100      32767      720176       12345678901
```

구문 도표에서, 용어 '정수'는 부호가 없어야 하는 큰 정수 상수에 대해 사용됩니다.

부동 소수점 상수

*부동 소수점 상수*는 부동 소수점 숫자를 F로 분리되는 두 숫자로 지정합니다. 첫 번째 수에는 부호와 소수점이 포함될 수 있고, 두 번째 수에서는 부호는 포함할 수 있지만 소수점은 포함할 수 없습니다. 부동 소수점 상수의 데이터 유형은 배정밀도입니다. 상수값은 두 번째 숫자에 의해 지정된 10의 승수와 첫번째 숫자로 산출한 것으로써, 부동 소수점 숫자 범위 내에 있어야 합니다. 상수에 있는 문자 수는 30을 초과해서는 안 됩니다.

예

```
15E1      2.E5      2.2E-1      +5.E+2
```

십진 상수

*십진 상수*는 부호가 있거나 또는 없는 숫자로서 최대 31 디지트로 구성되며, 소수점이 들어 있거나 2진 정수 범위 내에 있지 않습니다. 이것은 십진수 범위 내에 있어야 합니다. 정밀도는 디지트의 총 개수(앞에 오는 0과 뒤에 오는 0 포함)이며, 스케일은 소수점 오른쪽에 있는 디지트 개수(뒤에 오는 0 포함)입니다.

예

```
25.5     1000.     -15.     +37589.3333333333
```

리터럴

*문자열 상수*는 가변 길이 문자열을 지정하며, 작은 따옴표(')로 시작하고 끝나는 일련의 문자입니다. 이 양식의 리터럴은 문자열 분리 문자 사이에 있는 문자열을 지정합니다. 문자열의 길이는 32 672 바이트 이하여야 합니다. 두 개의 연속적인 문자열 분리 문자는 문자열 내에 있는 하나의 문자열 분리 문자를 나타내는 데 사용됩니다.

예

```
'12/14/1985'
'32'
'DON''T CHANGE'
```

동일하지 않은 코드 페이지 고려사항

상수값은 이것이 데이터베이스로 제한될 때 언제나 데이터베이스 코드 페이지로 변환됩니다. 이것은 데이터베이스 코드 페이지에 있는 것으로 간주됩니다. 그러므로, FOR BIT DATA 컬럼의 상수를 결합하는 표현식(이것의 결과도 FOR BIT DATA임)에서 사용되는 경우, 상수값은 사용된 데이터베이스 코드 페이지 형식으로 변환되지 *않습니다*.

16진 상수

*16진 상수*는 응용프로그램 서버(AS)의 코드 페이지를 갖는 가변 길이 문자열을 지정합니다.

16진 리터럴의 형식에서는 X 다음에 작은 따옴표(')로 시작하여 종료되는 일련의 문자가 옵니다. 작은 따옴표 사이에 있는 문자는 짝수 개의 16진 디지트여야 합니다. 16진 디지트 개수는 16 336을 초과할 수 없으며, 초과하는 경우 오류가 발생합니다(SQLSTATE -54002). 16진 디지트는 4 비트를 나타냅니다. 16진수는 숫자 또는 A에서 Z(대문자 또는 소문자)까지의 문자로 지정되며, 여기서 A는 비트 패턴 '1010'을 나타내며, B는 비트 패턴 '1011'을 나타냅니다. 16진 상수가 부적절하게 포맷된 경우(가령, 부당한 16진 디지트나 홀수 개의 16진 디지트가 포함된 경우), 오류가 발생됩니다(SQLSTATE 42606).

예

```
X'FFFF'       비트 패턴 '1111111111111111'
X'4672616E6B' ASCII 문자열 'Frank'의 VARCHAR 패턴에 해당됨.
```

그래픽 리터럴

*그래픽 문자열 상수*는 가변 길이 그래픽 문자열을 지정하며 1 바이트 작은 따옴표(')로 시작하고 끝나는 일련의 2 바이트 문자로 구성됩니다. 그리고, 1 바이트 G 또는 N이 앞에 붙습니다. 이러한 문자열 상수 양식은 문자열 분리문자 사이에 포함된 그래픽 문자열을 지정합니다. 그래픽 문자열의 길이는 짝수 바이트여야 하고, 16 336바이트보다 커서는 안 됩니다.

예:

```
G'2바이트 문자열'
N'2바이트 문자열'
```

MBCS에 관한 고려사항

작은 따옴표(')는 분리 문자로 간주되는 MBCS 문자의 일부로 표시되어서는 안 됩니다.

사용자 정의 유형의 상수 사용

사용자 정의 유형은 입력에 국한됩니다. 이것은 사용자 정의 유형이 자신의 유형과만 호환됨을 의미합니다. 그러나 상수에는 내장 유형이 있습니다. 그러므로, 사용자 정의 유형과 상수를 포함하는 연산은 사용자 정의 유형이 상수의 내장 유형으로 변환되거나 상수가 사용자 정의 유형으로 변환된 경우에만 가능합니다(유형 변환에 대해서는 201 페이지의 『유형변환(CAST) 스펙』에서 자세한 내용을 참조하십시오). 예를 들어, 123 페이지의 『사용자 정의 유형 비교』의 테이블과 구별 유형을 사용하는 경우, 상수 14를 사용한 다음 비교가 유효합니다.

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > CAST(14 AS YOUTH)

SELECT * FROM CAMP_DB2_ROSTER
   WHERE CAST(AGE AS INTEGER) > 14
```

다음 비교는 유효하지 않습니다.

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > 14
```

특수 레지스터

*특수 레지스터*는 데이터베이스 관리 프로그램이 응용프로그램 프로세스에 정의한 저장영역으로, SQL문에서 참조될 수 있는 정보를 보관하는 데 사용됩니다. 특수 레지스터는 데이터베이스 코드 페이지로 되어 있습니다.

CURRENT DATE

CURRENT DATE 특수 레지스터는 SQL문이 응용프로그램 서버(AS)에서 수행될 때의 시간을 기준으로 날짜를 지정합니다. 이 특수 레지스터가 하나의 SQL문에서 한번 이상 사용된 경우 또는 하나의 명령문에서 CURRENT TIME이나 CURRENT TIMESTAMP로 사용된 경우, 모든 값은 단일 시계 읽기를 기준으로 합니다.

연합 시스템에서는, 데이터 소스로 의도된 조회에서 CURRENT DATE를 사용할 수 있습니다. 조회가 처리될 때, 리턴되는 데이터는 데이터 소스가 아니라, 연합 서버에서 CURRENT DATE 레지스터로부터 확보됩니다.

예

PROJECT 테이블을 사용하여, MA2111 프로젝트(PROJNO)의 프로젝트 종료일(PRENDATE)을 현재 날짜로 설정하십시오.

```
UPDATE PROJECT
  SET PRENDATE = CURRENT DATE
  WHERE PROJNO = 'MA2111'
```

CURRENT DEFAULT TRANSFORM GROUP

CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터는 사용자가 정의하는 구조화 유형 값을 호스트 프로그램과 교환하기 위해 동적 SQL문에서 사용되는 변환 그룹의 이름을 식별하는 VARCHAR (18) 값을 지정합니다. 이 특수 레지스터는 정적 SQL문에서나, 매개변수와 결과를 메소드의 외부 함수들과 교환할 때 사용되는 변환 그룹을 지정하지 않습니다.

SET CURRENT DEFAULT TRANSFORM GROUP문으로 값을 설정할 있습니다. 어떤 값도 설정하지 않을 경우, 특수 레지스터의 초기 값은 빈 문자열(길이가 0인 VARCHAR)입니다.

동적 SQL문(즉, 호스트 변수와 상호작용하는 명령문)에서, 값 교환에 사용되는 변환 그룹의 이름은 이 레지스터에 빈 문자열이 포함되어 있지만 않으면 이 특수 레지스터의 값과 같습니다. 레지스터에 빈 문자열(SET CURRENT DEFAULT TRANSFORM GROUP문을 사용하여 설정된 값이 없음)이 있을 경우, DB2_PROGRAM 변환 그룹이 변환에 사용됩니다. DB2_PROGRAM 변환 그룹이 구조화 유형 주제에 대해 정의되지 않은 경우, 런타임 시 오류가 발생합니다 (SQLSTATE 42741).

예

기본 변환 그룹을 MYSTRUCT1으로 설정하십시오. MYSTRUCT1 변환에 정의된 TO SQL 및 FROM SQL 함수는 사용자가 정의하는 구조화 유형 변수를 호스트 프로그램과 교환하기 위해 사용됩니다.

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

이 특수 레지스터에 지정된 기본 변환 그룹 이름을 검색하십시오.

```
VALUES (CURRENT DEFAULT TRANSFORM GROUP)
```

CURRENT DEGREE

CURRENT DEGREE 특수 레지스터는 동적 SQL문의 실행에 대한 파티션 내 병렬 처리를 지정합니다.[18] 레지스터의 데이터 유형은 CHAR(5)입니다. 유효한 값은 'ANY' 또는 1에서 32 767까지의 정수 문자열입니다.

SQL문이 동적으로 준비되고 정수로 표시된 CURRENT DEGREE의 값이 1인 경우, 이 명령문의 실행은 파티션 내 병렬 처리를 사용하지 않습니다.

SQL문이 동적으로 준비된 경우 정수로 나타낸 CURRENT DEGREE의 값이 1보다 크거나 32 767 이하이면, 이 명령문의 실행에 지정된 등급의 파티션 내 병렬 처리가 적용됩니다.

SQL문이 동적으로 준비되고 CURRENT DEGREE의 값이 'ANY'인 경우, 이 명령문의 실행에는 데이터베이스 관리 프로그램이 정한 수준으로 파티션 내 병렬 처리가 관련됩니다.

---

18. 정적 SQL의 경우, DEGREE 바인드 옵션은 동일한 제어를 제공합니다.

병렬 처리의 실제 런타임 등급은 다음 등급보다 낮습니다.
- 최대 조회 수준(max_querydegree) 구성 매개변수
- 응용프로그램 런타임 수준
- SQL문 컴파일 수준

parallel_enable 데이터베이스 관리 프로그램 구성 매개변수가 NO로 설정된 경우, 최적화를 위해 CURRENT DEGREE 특수 레지스터의 값은 무시되며 명령문에는 파티션 내 병렬 처리가 사용되지 않습니다.

병렬 처리의 설명과 제한사항의 목록에 대해서는 *관리 안내서*에서 참조하십시오.

SET CURRENT DEGREE문을 실행하여 값을 변경할 수 있습니다. 이 명령문에 대해 1143 페이지의 『SET CURRENT DEGREE』에서 자세한 정보를 참조하십시오.

CURRENT DEGREE의 초기 값은 dft_degree 데이터베이스 구성 매개변수에 의해 결정됩니다. 이 구성 매개변수에 대해 *관리 안내서*에서 설명을 참조하십시오.

CURRENT EXPLAIN MODE

CURRENT EXPLAIN MODE 특수 레지스터는 CHAR(254) 값을 보유하며 이 값은 적합한 종료문에 대해 Explain 기능의 동작을 제어합니다. 이 기능은 Explain 정보를 생성하여 Explain 테이블에 이를 삽입합니다(자세한 사항은 *관리 안내서* 참조). 이 정보는 Explain 스냅샷에 포함되지 않습니다.

가능한 값은 YES, NO, EXPLAIN, RECOMMEND INDEXES 및 EVALUATE INDEXES입니다. [19]

YES Explain 기능을 작동시키고, 동적 SQL문에 대한 Explain 정보가 명령문 컴파일시 보관되도록 합니다.

EXPLAIN
YES와 같은 기능을 작동 가능하게 하지만, 동적 명령문이 실행되지 않습니다.

19. 정적 SQL의 경우, EXPLAIN 바인드 옵션은 동일한 제어를 제공합니다. PREP 및 BIND 명령의 경우, EXPLAIN 옵션 값은 YES, NO 및 ALL입니다.

NO 'Explain' 기능을 작동불가능으로 만듭니다.

RECOMMEND INDEXES
각 동적 조회에 대해, 색인들의 세트가 권장됩니다. ADVISE_INDEX 테이블은 색인들의 세트로 채워집니다.

EVALUATE INDEXES
동적 조회는 마치 권장 색인이 존재하듯이 설명됩니다. 색인들은 ADVISE_INDEX 테이블로부터 취해집니다.

초기값은 NO입니다.

CURRENT EXPLAIN MODE문으로 값을 변경할 수 있습니다. 이 명령문에 대해 1145 페이지의 『SET CURRENT EXPLAIN MODE』에서 정보를 참조하십시오.

CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터 값은 Explain 기능이 호출될 때 상호 작동합니다(자세한 내용은 1489 페이지의 표142 참조). CURRENT EXPLAIN MODE 특수 레지스터도 EXPLAIN 바인드 옵션을 사용하여 상호 작동합니다(자세한 내용은 1490 페이지의 표143 참조). 값 RECOMMEND INDEXES 및 EVALUATE INDEXES는 CURRENT EXPLAIN MODE 레지스터에 대해서만 설정될 수 있으므로, SET CURRENT EXPLAIN MODE문을 사용하여 설정되어야 합니다.

예: 호스트 변수 EXPL_MODE (VARCHAR(254))를 CURRENT EXPLAIN MODE 특수 레지스터의 현재 값으로 설정하십시오.

```
VALUES CURRENT EXPLAIN MODE
  INTO :EXPL_MODE
```

CURRENT EXPLAIN SNAPSHOT

CURRENT EXPLAIN SNAPSHOT 특수 레지스터는 CHAR(8) 값을 지정하며 이 값은 Explain 스냅샷 기능의 동작을 제어합니다. 이 기능은 액세스 플랜 정보, 조작원 비용 및 바인드시 통계를 포함한 압축된 정보를 생성합니다(자세한 정보는 *관리 안내서* 참조).

DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES 또는 VALUES INTO문은 해당 레지스터 값을 고려합니다.

가능한 값은 YES, NO 및 EXPLAIN입니다. [20]

YES 스냅샷 기능을 사용할 수 있으며, 명령문이 컴파일될 때, 동적 SQL문의 내부 표현에 대한 스냅샷을 취합니다.

EXPLAIN
YES와 같은 기능을 작동 가능하게 하지만, 동적 명령문이 실행되지 않습니다.

NO Explain 스냅샷 기능을 사용할 수 없습니다.

초기값은 NO입니다.

SET CURRENT EXPLAIN SNAPSHOT문으로 값을 변경할 수 있습니다(1148 페이지의 『SET CURRENT EXPLAIN SNAPSHOT』 참조).

CURRENT EXPLAIN SNAPSHOT 및 CURRENT EXPLAIN MODE 특수값은 Explain 기능이 호출될 때 상호 작동합니다(자세한 설명은 1489 페이지의 표 142 참조). CURRENT EXPLAIN SNAPSHOT 특수 레지스터도 EXPLSNAP 바인드 옵션을 사용하여 상호 작동합니다(자세한 설명은 1491 페이지의 표144 참조).

예
호스트 변수 EXPL_SNAP (char(8))를 CURRENT EXPLAIN SNAPSHOT 특수 레지스터의 현재 값으로 설정하십시오.

```
VALUES CURRENT EXPLAIN SNAPSHOT
  INTO :EXPL_SNAP
```

CURRENT NODE

CURRENT NODE 특수 레지스터는 조정자(coordinator) 노드 번호(응용프로그램이 접속하는 파티션)를 식별하는 INTEGER 값을 지정합니다.

---

20. 정적 SQL의 경우, EXPLSNAP 바인드 옵션은 동일한 제어를 제공합니다. PREP 및 BIND 명령의 경우, EXPLSNAP 옵션 값은 YES, NO 및 ALL입니다.

데이터베이스 인스턴스가 파티션을 지원하도록 정의되지 않은 경우 CURRENT NODE는 0을 리턴합니다(db2nodes.cfg 파일 없음).[21] .

CURRENT NODE는 CONNECT문으로 변경할 수 있지만, 특정 조건하에서만 가능합니다(608 페이지의 『CONNECT(유형 1)』 참조).

예

호스트 변수 APPL_NODE(정수)를 응용프로그램이 연결된 파티션의 이름으로 설정하십시오.

```
VALUES CURRENT NODE
  INTO :APPL_NODE
```

CURRENT PATH

CURRENT PATH 특수 레지스터는 동적으로 준비된 SQL문에 사용되는 함수 참조와 데이터 유형 참조를 해석하는 데 사용될 *SQL 경로*를 식별하는 VARCHAR(254) 값을 지정합니다.[22] CURRENT PATH는 CALL문 내의 저장 프로시듀어 호출을 해석하는 데도 사용됩니다. 초기값은 아래에 지정된 기본값입니다. 정적 SQL의 경우, FUNCPATH 바인드 옵션은 함수 및 데이터 유형 분석에 사용되는 SQL 경로를 제공합니다(FUNCPATH 바인드 옵션에 대한 자세한 사항은 *Command Reference* 참조).

CURRENT PATH 특수 레지스터에는 하나 이상의 스키마 이름 목록이 들어 있습니다(여기서, 스키마 이름은 큰 따옴표로 묶여 있고 쉼표로 분리됩니다. 문자열 내의 모든 따옴표는 이들이 분리 식별자에 있는 것만큼 반복됩니다.).

예를 들어, SQL 경로가 데이터베이스 관리 프로그램이 FERMAT, XGRAPHIC을 차례로 검사하도록 하면, CURRENT PATH 특수 레지스터에서 SYSIBM 스키마가 다음과 같이 반환됩니다.

```
"FERMAT","XGRAPHIC","SYSIBM"
```

---

21. 파티션된 데이터베이스의 경우, db2nodes.cfg 파일이 존재하며 이 파일에 파티션(또는 노드) 정의가 포함되어 있습니다. *관리 안내서*에서 자세한 내용을 참조하십시오.

22. CURRENT FUNCTION PATH는 CURRENT PATH의 동의어입니다.

기본값은 "SYSIBM", "SYSFUN", X입니다. 여기서 X는 큰 따옴표로 분리되는 USER 특수 레지스터의 값입니다.

SET CURRENT FUNCTION PATH문으로 값을 변경할 수 있습니다(1174 페이지의 『SET PATH』 참조). 스키마 SYSIBM은 지정될 스키마 SYSIBM은 지정될 필요가 없습니다. SQL 경로에 포함되지 않을 경우, 이것은 내재적으로 첫번째 스키마로서 간주됩니다. SYSIBM에서는 내재적으로 가정된 경우에 254 문자 중 어떠한 문자도 취하지 않습니다.

함수 차수시 SQL 경로를 사용하는 방법이 165 페이지의 『함수』에 설명되어 있습니다. 스키마 이름으로 규정되지 않은 데이터 유형은 SQL 경로에 있는 최초의 스키마 이름으로 내재적으로 규정되며, 지정된 같은 규정화되지 않은 이름의 데이터 유형을 포함합니다. CREATE DISTINCT TYPE, CREATE FUNCTION, COMMENT ON 및 DROP문에서 설명된 대로 이 규칙에 예외가 있습니다.

예

SYSCAT.VIEWS 카탈로그 뷰를 사용하여, CURRENT PATH 특수 레지스터의 현재 값과 동일한 설정으로 작성된 모든 뷰를 찾으십시오.

```
SELECT VIEWNAME, VIEWSCHEMA FROM SYSCAT.VIEWS
  WHERE FUNC_PATH = CURRENT PATH
```

CURRENT QUERY OPTIMIZATION

CURRENT QUERY OPTIMIZATION 특수 레지스터는 INTEGER값을 지정하며, 이것은 동적 SQL문을 바인드 할때 데이터베이스 관리 프로그램에서 수행된 조회 최적화 클래스를 제어합니다. QUERYOPT 바인드 옵션은 정적 SQL문의 조회 최적화를 위한 클래스를 제어합니다(QUERYOPT 바인드 옵션에 대한 자세한 사항은 *Command Reference* 참조). 가능한 값은 0에서 9까지입니다. 예를 들어, 조회 최적화 클래스가 최적화의 최저 클래스(0)로 설정되면, 특수 레지스터에 있는 값은 0입니다. 기본값은 dft_queryopt 데이터베이스 구성 매개변수에 의해 결정됩니다.

SET CURRENT QUERY OPTIMIZATION문으로 값을 변경할 수 있습니다(1152 페이지의 『SET CURRENT QUERY OPTIMIZATION』 참조).

예

SYSCAT.PACKEGES 카탈로그 값을 사용하여, CURRENT QUERY OPTIMIZATION 특수 레지스터의 현재 값과 같은 설정치로 제한된 모든 계획을 찾습니다.

```
SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

CURRENT REFRESH AGE

CURRENT REFRESH AGE 특수 레지스터는 데이터 유형이 DECIMAL(20,6)인 시간소인 기간 값을 지정합니다. 이 기간은 REFRESH DEFERRED 요약 테이블 (REFRESH TABLE문이 조회 처리를 최적화하는 데 사용될 수 있는 요약 테이블)에서 처리되기 때문에 최대 기간입니다. CURRENT REFRESH AGE의 값이 99 999 999 999 999(ANY)이고 QUERY OPTIMIZATION 클래스가 5이상이면, REFRESH DEFERRED 요약 테이블은 동적 SQL 조회의 처리를 최적화할 것으로 생각됩니다. REFRESH IMMEDIATE 속성을 가지며 점검 보류 상태에 있지 않은 요약 테이블은 화면갱신 간격이 0입니다.

값은 SET CURRENT REFRESH AGE문으로 변경할 수 있습니다(1156 페이지의『SET CURRENT REFRESH AGE』참조). REFRESH DEFERRED에 정의된 요약 테이블은 정적 embedded SQL 조회로 간주되지 않습니다.

CURRENT REFRESH AGE의 초기 값은 0입니다.

CURRENT SCHEMA

CURRENT SCHEMA 특수 레지스터는 동적으로 준비된 SQL문에서 적용할 수 있는 규정되지 않은 데이터베이스 오브젝트 참조를 규정하기 위해 사용되는 스키마 이름을 식별하는 VARCHAR(128) 값을 지정합니다. [23]

CURRENT SCHEMA의 초기값은 현재 세션 사용자의 권한 부여 ID입니다.

값은 SET SCHEMA문에 의해 변경될 수 있습니다(1177 페이지의『SET SCHEMA』참조).

---

23. OS/390용 DB2와의 호환을 위해, 특수 레지스터 CURRENT SQLID는 CURRENT SCHEMA와 동의어로 간주됩니다.

QUALIFIER 바인드 옵션은 규정화되지 않은 데이터베이스 오브젝트 참조(정적 SQL문에 적용 가능)를 규정하는 데 사용되는 스키마 이름을 제어합니다(*Command Reference* 참조).

예

오브젝트 규정화에 대한 스키마를 'D123'으로 설정하십시오.

```
SET CURRENT SCHEMA =  'D123'
```

CURRENT SERVER

CURRENT SERVER 특수 레지스터는 현재의 응용프로그램 서버(AS)를 나타내는 VARCHAR(18)값을 지정합니다. 응용프로그램 서버의 실제 이름(별명이 아님)이 레지스터에 포함됩니다.

CURRENT SERVER는 CONNECT문으로만 변경될 수 있지만 일정한 조건으로 제한됩니다(608 페이지의 『CONNECT(유형 1)』 참조).

예

호스트 변수 APPL_SERVE(VARCHAR(18))를 응용프로그램이 연결된 응용프로그램 서버(AS)의 이름으로 설정하십시오.

```
VALUES CURRENT SERVER
  INTO :APPL_SERVE
```

CURRENT TIME

CURRENT TIME 특수 레지스터는 SQL문이 응용프로그램 서버(AS)에서 수행될 때의 시간을 기준으로 시간을 지정합니다. 이 특수 레지스터가 하나의 SQL문에서 한번 이상 사용된 경우 또는 하나의 명령문에서 CURRENT DATE나 CURRENT TIMESTAMP와 함께 사용된 경우, 모든 값은 단일 시계 읽기를 기준으로 합니다.

연합 시스템에서는, 데이터 소스로 의도된 조회에서 CURRENT TIME을 사용할 수 있습니다. 조회가 처리될 때, 리턴되는 시간은 데이터 소스가 아니라, 연합 서버에서 CURRENT TIME 레지스터로부터 확보됩니다.

예

CL_SCHED 테이블을 사용하여, 오늘 나중에 시작하는 모든 클래스(CLASS_CODE)를 선택하십시오. DAY 컬럼에 있는 오늘의 클래스에는 3값이 있습니다.

```
SELECT CLASS_CODE FROM CL_SCHED
  WHERE STARTING > CURRENT TIME  AND DAY = 3
```

CURRENT TIMESTAMP

CURRENT TIMESTAMP 특수 레지스터는 SQL문이 응용프로그램 서버(AS)에서 수행될 때의 시간을 기준으로 시간을 지정합니다. 이 특수 레지스터가 하나의 SQL문에서 한번 이상 사용된 경우 또는 단일의 명령문에서 CURRENT DATE나 CURRENT TIME과 함께 사용된 경우, 모든 값은 하나의 시계 읽기를 기준으로 합니다.

연합 시스템에서는, 데이터 소스로 의도된 조회에서 CURRENT TIMESTAMP를 사용할 수 있습니다. 조회가 처리될 때, 리턴되는 시간소인은 데이터 소스가 아니라, 연합 서버에서 CURRENT TIMESTAMP 레지스터로부터 확보됩니다.

예

IN_TRAY 테이블에 한 행을 삽입합니다. RECEIVED 컬럼의 값은 행이 삽입된 때를 나타내는 시각 값이어야 합니다. 다음 세 가지 컬럼에 대한 값은 호스트 변수 SRC(char(8)), SUB(char(64)) 및 TXT(VARCHAR(200))에서 구한 값입니다.

```
INSERT INTO IN_TRAY
  VALUES (CURRENT TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT TIMEZONE

CURRENT TIMEZONE 특수 레지스터는 UTC[24]와 응용프로그램 서버(AS)의 지역 시간과의 차이를 지정합니다. 차이는 시간 기한(십진수로서, 처음 두 디지트는 시간, 다음 두 디지트는 분, 마지막 두 디지트는 초입니다)으로 표현됩니다. 시간

24. 국제 표준시(Coordinated Universal Time), 일반적으로 GMT로 알려짐

의 숫자는 -24와 24 이내입니다. 지역 시간에서 CURRENT TIMEZONE을 빼면 UTC로 지역 시간이 변형됩니다. 시간은 SQL문이 수행될 때 운영 시스템의 시간에서 계산됩니다.[25]

CURRENT TIMEZONE 특수 레지스터는 십진수(6,0) 데이터 유형 표현식이 사용되는 곳이면 어디에서나 사용될 수 있습니다(예를 들면, 시간과 시간소인 연산).

예

레코드를 RECEIVED 컬럼에 대한 UTC 시각을 사용하여 IN_TRAY 테이블로 삽입합니다.

```
INSERT INTO IN_TRAY VALUES(
       CURRENT TIMESTAMP - CURRENT TIMEZONE,
       :source,
       :subject,
       :notetext )
```

USER

USER 특수 레지스터는 응용프로그램이 데이터베이스에서 시작될 때 데이터베이스 관리 프로그램으로 전달된 런타임 권한 부여 ID를 지정합니다. 레지스터의 데이터 유형은 VARCHAR(128)입니다.

예

IN_TRAY 테이블에서 사용자가 스스로 위치시킨 모든 참고를 선택하십시오.

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = USER
```

컬럼 이름

*컬럼 이름*의 뜻은 구문에 따라 다릅니다. 컬럼 이름은 다음에 사용될 수 있습니다.

- CREATE TABLE문에서, 컬럼 이름을 선언하기 위해
- CREATE INDEX문에서, 컬럼을 나타내기 위해

25. CURRENT TIMEZONE 값은 C 런타임 함수에서 결정합니다. 시간대에 관한 설치 요구사항에 대한 *빠른 시작*에서 정보를 참조하십시오.

- 다음 구문에서처럼, 컬럼값을 지정하기 위해
  - 컬럼 함수에서, 컬럼 이름은 함수가 적용되는 그룹이나 임시 결과 테이블에 있는 컬럼의 모든 값을 지정합니다(그룹과 임시 결과 테이블은 433 페이지의 『제5장 조회』에서 설명합니다.). 예를 들어, MAX(SALARY)는 함수 MAX를 그룹에 있는 컬럼 SALARY의 모든 값에 적용합니다.
  - GROUP BY나 ORDER BY절에서, 컬럼 이름은 절이 적용되는 임시 결과 테이블에 있는 모든 값을 지정합니다. 예를 들어, ORDER BY DEPT는 컬럼 DEPT의 값에 의해 임시 결과 테이블의 순서를 지정합니다.
  - 표현식, 검색 조건 또는 스칼라 함수에서, 컬럼 이름은 구조가 적용된 각 행이나 그룹에 대한 값을 지정합니다. 예를 들어, 검색 조건 CODE = 20이 일부 행에 적용되면, 컬럼 이름 CODE가 지정한 값은 그 행에 있는 컬럼 CODE의 값입니다.
- FROM절의 *테이블 참조*에 있는 *상관 절*에서와 같이 일시적으로 컬럼을 재명명하십시오.

규정화된 컬럼 이름

컬럼 이름에 대한 규정자는 테이블, 뷰, 별명(nickname), 별명(alias) 또는 상관 이름이 될 수 있습니다.

컬럼 이름이 규정될 수 있는지의 여부는 구문에 따라 다릅니다.

- COMMENT ON문의 양식에 따라, 단일 컬럼 이름은 규정화되어야 합니다. 중복 컬럼 이름은 규정화될 수 없습니다.
- 컬럼 이름이 컬럼값을 지정하는 것은 사용자 옵션에서 규정화됩니다.
- 다른 모든 구문에서, 컬럼 이름은 규정화될 수 없습니다.

규정자가 옵션인 것은 두 가지 목적을 제공됩니다. 이것에 대해서는 151 페이지의 『모호성을 회피하기 위한 컬럼 이름 규정자』 및 154 페이지의 『상관 참조에서의 컬럼 이름 규정자』에서 설명합니다.

상관 이름

*상관 이름*은 조회의 FROM절과, UPDATE나 DELETE문의 처음 절에서 정의될 수 있습니다. 예를 들어, FROM X.MYTABLE Z절에서는 X.MYTABLE에 대한 상관 이름으로 Z를 설정합니다.

```
FROM X.MYTABLE Z
```

X.MYTABLE에 대한 상관 이름이 Z로 정의되면 Z만이 SELECT문에서 X.MYTABLE 인스턴스의 컬럼을 참조하도록 규정하는 데 사용될 수있습니다.

상관 이름은 정의된 문맥 내에서만 테이블, 뷰, 별명, 중첩 테이블 표현식 또는 테이블 함수와 연관됩니다. 그러므로, 같은 상관 이름은 다른 명령문에서 또는 같은 명령문 다른 절(clause)에서 다른 목적으로 정의될 수 있습니다.

규정자로서, 상관 이름은 모호성을 피하거나 상관된 참조를 구축하는 데 사용될 수 있습니다. 이 이름은 테이블, 뷰, 별명(nickname), 별명(alias)의 단축 이름으로서만 사용될 수 있습니다. 중첩 테이블 표현식 또는 테이블 함수의 경우, 결과 테이블을 식별하기 위해 상관 이름이 필요합니다. 이 예에서, Z는 한번 이상 X.MYTABLE을 입력하지 않도록 사용되었습니다.

테이블, 뷰, 별명(nickname) 또는 별명(alias) 이름에 대해 상관 이름이 지정될 경우, 테이블, 뷰, 별명(nickname) 또는 별명(alias) 이름의 해당되는 인스턴스 컬럼에 대한 규정 참조에는 테이블, 뷰, 별명(nickname) 또는 별명(alias) 이름보다는, 상관 이름이 사용되어야 합니다. 예를 들어, 다음 예에서 EMPLOYEE.PROJECT에 대한 참조는 올바르지 않습니다. EMPLOYEE에 대해 상관 이름이 지정되었기 때문입니다.

예

```
FROM EMPLOYEE E
WHERE EMPLOYEE.PROJECT='ABC'      * incorrect*
```

PROJECT에 대한 규정된 참조는 아래에 표시된 것처럼 상관 이름 ″E″를 대신 사용해야 합니다.

```
FROM EMPLOYEE E
  WHERE E.PROJECT='ABC'
```

FROM절에 지정된 이름은 노출되었거나 *비*노출되었습니다. 상관 이름을 지정하지 않으면 테이블, 뷰, 별명(nickname) 또는 별명(alias) 이름이 FROM 절에서 노출되어 있다고 말할 수 있습니다. 상관 이름은 언제나 표시 이름입니다. 예를 들어, 다음 FROM 절에서, 상관 이름은 DEPARTMENT가 아니라 EMPLOYEE에 대해 지정되므로, DEPARTMENT는 노출된 이름이고 EMPLOYEE는 그렇지 않습니다.

```
FROM EMPLOYEE E, DEPARTMENT
```

FROM 절에서 노출된 테이블, 뷰, 별명(nickname) 또는 별명(alias) 이름은 그 FROM 절에서 노출된 다른 테이블 이름, 뷰 이름 또는 별명(nickname)과 같거나, FROM 절에 있는 상관 이름이 될 수 있습니다. 이것은 모호한 컬럼 이름 참조로 인한 오류(SQLSTATE 42702)가 발생할 수 있습니다.

다음에 표시된 처음 두 FROM절은 맞습니다. 이것은 각각에 노출된 EMPLOYEE로 한번 이상 참조하지 않기 때문입니다.

1. FROM절에서,

   ```
   FROM EMPLOYEE E1, EMPLOYEE
   ```

   EMPLOYEE.PROJECT 같은 규정 참조는 FROM절에서 EMPLOYEE의 두 번째 인스턴스의 컬럼을 표시합니다. EMPLOYEE의 첫번째 인스턴스로의 규정된 참조는 상관 이름 “E1”(E1.PROJECT)를 사용해야 합니다.

2. FROM절에서,

   ```
   FROM EMPLOYEE, EMPLOYEE E2
   ```

   EMPLOYEE.PROJECT 같은 규정 참조는 FROM절에서 EMPLOYEE의 첫 번째 인스턴스의 컬럼을 표시합니다. EMPLOYEE의 두 번째 인스턴스로의 규정된 참조는 상관 이름 “E2”(E2.PROJECT)을 사용해야 합니다.

3. FROM절에서,

   ```
   FROM EMPLOYEE, EMPLOYEE
   ```

   이 절에 포함된 두 개의 노출된 테이블 이름(EMPLOYEE와 EMPLOYEE)이 동일합니다. 동일한 이름은 허용되지만 특정 컬럼 이름을 참조하는 것은 보호할 수 있습니다(SQLSTATE 42702).

4. 다음 명령문에서,

```
SELECT *
FROM EMPLOYEE E1, EMPLOYEE E2              * incorrect *
  WHERE EMPLOYEE.PROJECT = 'ABC'
```

   규정된 참조 EMPLOYEE.PROJECT는 잘못되었습니다. 이것은 FROM절에 있는 EMPLOYEE의 인스턴스 모두에 상관 이름이 있기 때문입니다. 대신, PROJECT의 참조는 상관 이름(E1.PROJECT 또는 E2.PROJECT)으로 규정화되어야 합니다.

5. FROM절에서,

```
FROM EMPLOYEE, X.EMPLOYEE
```

   EMPLOYEE의 두 번째 인스턴스에 있는 컬럼의 참조는 X.EMPLOYEE(X.EMPLOYEE.PROJECT)를 사용해야 합니다. 동적 SQL 또는 정적 SQL의 QUALIFIER precompile/bind 옵션의 CURRENT SCHEMA 특수 레지스터 값이 X일 경우, 이러한 참조는 애매하기 때문에 컬럼을 참조할 수 없습니다.

FROM절에서 상관 이름을 사용하면, 컬럼 이름 목록을 결과 테이블의 컬럼과 연관되게 지정하는 옵션을 사용할 수 있습니다. 상관 이름에서와 마찬가지로, 나열되는 이러한 이름들이 조회 전반에 걸쳐 컬럼에 대한 참조로 사용되어야 하는 컬럼의 *노출된* 이름이 됩니다. 컬럼 이름 목록이 지정되는 경우, 내재하는 결과 테이블의 컬럼 이름은 *비노출형* 이름이 됩니다.

FROM절에서,

```
FROM DEPARTMENT D(NUM,NAME,MGR,ANUM,LOC)
```

D.NUM과 같은 규정된 참조는 테이블에서 DEPTNO로 정의된 DEPARTMENT 테이블의 첫번째 컬럼을 나타냅니다. 이러한 FROM절을 사용한 D.DEPTNO에 대한 참조는 컬럼 이름 DEPTNO가 비노출형 컬럼 이름이므로 부정확합니다.

모호성을 회피하기 위한 컬럼 이름 규정자

함수, GROUP BY 절, ORDER BY 절, 표현식 또는 검색 조건 문맥에서, 컬럼 이름은 일부 테이블, 뷰, 별명(nickname), 중첩 테이블 표현식 또는 테이블 함수에서 컬럼의 값을 참조합니다. 컬럼을 포함할 수 있는 테이블, 뷰, 별명(nickname), 중첩 테이블 표현식 및 테이블을 문맥의 *오브젝트 테이블*이라고 합니다. 둘 이상

의 오브젝트 테이블에는 같은 이름의 컬럼이 있을 수 있습니다. 컬럼 이름을 규정하는 한가지 이유는 컬럼을 추출할 테이블을 지정하기 위해서입니다. 컬럼 이름의 규정자는 또한 SQL문에서 사용되는 SQL 변수 이름을 컬럼 이름과 구별하기 위한 SQL 프로시듀어에서 유용합니다.

중첩 테이블 표현식이나 테이블 함수는 FROM 절에서 선행하는 *테이블 참조*를 오브젝트 테이블로 간주합니다. 뒤에 오는 *테이블 참조*는 오브젝트 테이블로 간주되지 않습니다.

테이블 지정자

특정 오브젝트 테이블을 지시하는 규정자를 *테이블 지정자*라고 합니다. 오브젝트 테이블을 나타내는 절에서는 이들에 대한 테이블 지정자도 구축합니다. 예를 들어, SELECT절에 있는 표현식의 오브젝트 테이블은 FROM절에서 다음과 같이 이름이 지정됩니다.

```
SELECT CORZ.COLA, OWNY.MYTABLE.COLA
  FROM OWNX.MYTABLE CORZ, OWNY.MYTABLE
```

FROM 절의 테이블 지정자는 다음과 같이 설정됩니다.

- 테이블, 뷰, 별명(nickname), 별명(alias), 중첩 테이블 표현식 또는 테이블 함수 뒤에 나오는 이름은 상관 이름과 테이블 지정자입니다. 그러므로, CORZ는 테이블 지정자입니다. CORZ는 선택 목록에 있는 첫번째 컬럼 이름을 규정하는 데 사용됩니다.
- 노출된 테이블, 뷰 이름, 별명(nickname) 또는 별명(alias)은 테이블 지정자입니다. 그러므로, OWNY.MYTABLE은 테이블 지정자입니다. OWNY.MYTABLE은 선택 목록에 있는 두 번째 컬럼 이름을 규정하는 데 사용됩니다.

각 테이블 지정자는 특정 FROM절에서 고유하게 하여, 컬럼의 모호한 참조의 가능성을 없애야 합니다.

미정의 및 모호한 참조 예방

컬럼 이름이 컬럼의 값을 참조할 때, 정확히 하나의 오브젝트 테이블이 그 이름의 컬럼을 포함해야 합니다. 다음 상황은 오류로 간주됩니다.

- 지정된 이름의 컬럼이 들어 있는 오브젝트 테이블이 없습니다. 참조가 정의되지 않았습니다.

- 컬럼 이름이 테이블 지정자에 의해 규정되었지만, 지정된 테이블에 지정된 이름의 컬럼이 없습니다. 또다시 참조가 정의되지 않았습니다.
- 이름이 규정화되지 않았고, 하나 이상의 오브젝트 테이블에 그 이름의 컬럼이 있습니다. 참조가 모호합니다.
- 컬럼 이름이 테이블 지정자에 의해 규정화되었지만, 지정된 테이블이 FROM절에서 고유하지 않고, 지정된 테이블의 반복이 컬럼에 있습니다. 참조가 모호합니다.
- 컬럼 이름이 TABLE 키워드가 선행하지 않는 중첩 테이블 표현식안에 있거나, 오른쪽 외부 조인 또는 완전 외부 조인의 오른쪽 피연산자인 중첩된 테이블 표현식안에 또는 테이블 기능안에 있습니다. 이 컬럼 이름은 중첩된 테이블 표현식의 full select 내 *테이블-참조(table-reference)* 컬럼에 언급하고 있습니다. 참조가 정의되지 않았습니다.

컬럼 이름을 테이블 지정자에서 고유하게 정의함으로써 모호함을 예방할 수 있습니다. 컬럼이 다른 이름의 여러 오브젝트 테이블에 포함된 경우, 테이블 이름이 지정자로 사용될 수 있습니다. 상관 이름에 후속하는 컬럼 이름 목록을 사용하여 한 오브젝트 테이블의 컬럼에 고유한 이름을 제공함으로써 테이블 지정자를 사용하지 않음에 따른 모호한 참조를 피할 수도 있습니다.

테이블 지정자의 노출된 테이블 이름 양식으로 컬럼을 규정할 때, 규정되거나 규정되지 않은 양식의 노출된 테이블 이름이 모두 사용됩니다. 그러나, 사용되는 규정자와 사용되는 테이블은 테이블 이름, 뷰 이름 또는 별명(nickname)과 테이블 지정자를 완전하게 규정하고 난 후와 같아야 합니다.

1. 명령문의 권한 부여 ID가 CORPDATA인 경우,

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
                          FROM EMPLOYEE
```

   유효한 명령문입니다.

2. 명령문의 권한 부여 ID가 REGION인 경우,

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
  FROM EMPLOYEE                          * incorrect *
```

유효하지 않습니다. 이는 EMPLOYEE가 테이블 REGION.EMPLOYEE를 나타내지만 WORKDEPT에 대한 규정자는 다른 테이블 CORPDATA.EMPLOYEE를 나타내기 때문입니다.

상관 참조에서의 컬럼 이름 규정자

*fullselect*는 다양한 SQL문의 구성요소로 사용될 수 있는 조회의 양식입니다. fullselects에 대한 433 페이지의 『제5장 조회』에서 좀더 자세한 정보를 참조하십시오. 명령문의 검색 조건에서 사용된 fullselect를 *부속 조회(subquery)*라고 합니다. 단일 값을 검색하기 위해 단일 명령문내의 표현식으로 사용되는 fullselect를 *스칼라 fullselect* 또는 *스칼라 부속 조회*라 합니다. 조회의 FROM절에 사용된 fullselect를 *중첩 테이블 표현식*이라 합니다. 검색 조건의 부속 조회, 스칼라 부속 조회 및 중첩 테이블 표현식은 이 주제항목의 나머지 부분에서 부속 조회로 지칭됩니다.

부속 조회 자체에 부속 조회가 포함될 수 있으며, 이 안에 다시 부속 조회가 포함될 수 있습니다. 그러므로, SQL문에는 부속 조회의 계층이 들어 있습니다. 부속 조회가 들어 있는 계층의 요소는 이들이 포함된 부속 조회보다 상위 레벨에 있다고 합니다.

계층의 모든 요소에는 하나 이상의 지정자가 있습니다. 부속 조회는 계층에서 자신 레벨에 있는 테이블의 컬럼뿐만 아니라 계층에서 이전에 식별된 테이블의 컬럼까지 계층의 상위 레벨로 참조합니다. 상위 레벨에서 식별된 테이블의 컬럼 참조를 *상관 참조(correlated reference)*라고 합니다.

SQL에 대한 기존 표준과의 호환성을 위해, 규정된 컬럼 이름과 규정되지 않은 컬럼 이름 모두가 상관 참조로 허용됩니다. 그러나, 부속 조회에 사용된 모든 컬럼 참조를 규정하는 것이 바람직하며, 그렇게 하지 않을 경우, 동일한 컬럼 이름으로 인해 의도하지 않은 결과가 나올 수 있습니다. 예를 들어, 한 계층에 있는 테이블이 상관 참조와 같은 컬럼 이름이 들어 있도록 변경되고, 명령문이 다시 준비되는 경우, 참조는 변경된 테이블에 적용됩니다.

부속 조회에 있는 컬럼 이름이 규정화되면, 계층의 각 레벨이 규정화된 컬럼 이름이 나타나는 같은 부속 조회에서 시작하여 규정자와 일치하는 테이블 지정자를 찾을 때까지 계층의 상위 레벨로 계속 검색됩니다. 일단 발견되면, 테이블이 주어진

컬럼을 가지고 있는지 확인합니다. 테이블이 컬럼 이름이 들어 있는 레벨보다 상위에서 발견된 경우, 이것은 테이블 참조가 발견된 레벨의 상관 참조입니다. 중첩 테이블 표현식의 fullselect 상위 계층을 검색하려면, 선택적 TABLE 키워드가 중첩 테이블 표현식에 선행되어야 합니다.

부속 조회에 있는 컬럼 이름이 규정화되지 않으면, 계층의 각 레벨에서 참조된 테이블이, 컬럼 이름이 나타나는 같은 부속 조회에서 시작하여 일치하는 컬럼 이름을 찾을 때까지 계층의 상위 레벨로 계속 검색됩니다. 컬럼을 컬럼 이름이 들어 있는 레벨보다 상위 레벨에 있는 테이블에서 발견한 경우, 컬럼이 들어 있는 테이블이 발견된 레벨의 상관 참조입니다. 컬럼 이름이 특정 레벨에서 하나 이상의 테이블에서 발견되는 경우, 참조는 모호해지고, 오류로 간주됩니다.

이경우, 다음 예에서 사용된 T는 컬럼 C가 들어 있는 테이블 지정자를 참조합니다. 컬럼 이름 T.C(여기서, T는 내재적 또는 표시된 규정자를 나타냅니다.)는 이러한 조건을 만족할 경우에만 상관 참조가 됩니다.

- T.C는 부속 조회의 표현식에 사용됩니다.
- T는 부속 조회의 FROM절에서 사용된 테이블을 지정하지 않습니다.
- T는 부속 조회가 들어 있는 계층의 상위 레벨에서 사용된 테이블을 지정합니다.

같은 테이블, 뷰 또는 별명(nickname)을 여러 레벨에서 식별할 수 있으므로, 테이블 지정자로 고유한 상관 이름을 사용하는 것이 좋습니다. 하나 이상 레벨에서 테이블을 지시하기 위해 T가 사용될 경우(T는 그 자체가 테이블 이름이거나 중복되는 상관 이름입니다.), T.C는 대부분 T.C를 포함하는 부속 조회를 직접 포함하는 T가 사용되는 레벨을 참조합니다. 상위 레벨로의 상관이 필요할 경우, 고유한 상관 이름을 사용해야 합니다.

상관 참조 T.C는 검색 조건이 적용되는 행이나 그룹 T에 있는 값 C를 나타내며, 조건 1은 부속 조회에, 조건 2는 상위 레벨로 지정합니다. 조건 2가 WHERE절에서 사용되는 경우, 부속 조회는 조건 2가 적용되는 각 행에 대해 평가됩니다. 조건 2가 HAVING절에서 사용되는 경우, 부속 조회는 조건 2가 적용되는 각 행에 대해 평가됩니다(부속 조회의 평가에 대해서는, 433 페이지의 『제5장 조회』에 있는 WHERE와 HAVING절의 설명을 참조하십시오.).

예를 들어, 다음 명령문에서, 상관 참조 X.WORKDEPT(마지막 행에 있음)는 첫 번째 FROM절 레벨에 있는 테이블 EMPLOYEE의 WORKDEPT 값을 참조합니다. (이 절은 X를 EMPLOYEE에 대한 상관 이름으로 설정합니다.) 명령문에서는 그 부서의 평균 봉급보다 봉급이 작은 사원을 열거합니다.

```
SELECT EMPNO, LASTNAME, WORKDEPT
  FROM EMPLOYEE X
  WHERE SALARY <(SELECT AVG(SALARY)
                           FROM EMPLOYEE
                    WHERE WORKDEPT = X.WORKDEPT)
```

다음 예에서는 상관 이름으로 THIS를 사용합니다. 명령문은 사원이 없는 부서에 대해 행을 삭제합니다.

```
DELETE FROM DEPARTMENT THIS
   WHERE NOT EXISTS(SELECT *
                           FROM EMPLOYEE
                      WHERE WORKDEPT = THIS.DEPTNO)
```

호스트 변수 참조

호스트 *변수*는 다음 중 하나입니다.

- C 변수, C++ 변수, COBOL 데이터 항목, FORTRAN 변수 또는 Java 변수와 같은 호스트 언어에서의 변수.

또는

- SQL문에서 참조되는 SQL 확장으로 선언된 변수에서 SQL 사전 처리 컴파일러에 의해 작성된 호스트 언어 구성

호스트 변수는 호스트 언어에 있는 명령문으로 직접 정의되거나 SQL 확장을 사용하여 간접적으로 정의됩니다.

SQL문에 있는 호스트 변수는 호스트 변수 선언 규칙에 따라 프로그램에서 설명한 호스트 변수를 식별해야 합니다.

SQL문에 사용되는 모든 호스트 변수는 REXX를 제외한 모든 호스트 언어로 SQL DECLARE 섹션에 정의되어야 합니다(*응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.). SQL DECLARE절에서 선언된 변수와 같은 이름으로 SQL

DECLARE절 외부에서 변수를 선언할 수 없습니다. SQL DECLARE절은 BEGIN DECLARE SECTION으로 시작하여 END DECLARE SECTION으로 종료합니다.

구문 도표에서 사용된 메타-변수 *호스트-변수*는 호스트 변수에 대한 참조를 보여줍니다. VALUES INTO절이나, FETCH 또는 SELECT INTO문의 INTO절에 있는 *호스트-변수*는 행의 컬럼 또는 표현식의 값이 할당될 *호스트-변수*를 식별합니다. 기타 모든 구문에서, 호스트 변수는 응용프로그램에서 데이터베이스 관리 프로그램으로 전달된 값을 지정합니다.

동적 SQL에서의 호스트 변수

동적 SQL문에서, 매개변수 표시문자는 호스트 변수 대신 사용됩니다. 매개변수 표시문자는 물음표(?)로서 응용프로그램이 값을 제공하는 동적 SQL에서의 위치를 나타냅니다. 즉, 명령문 문자열이 정적 SQL문인 경우 호스트 변수가 발견되는 위치를 나타냅니다. 다음 예는 호스트 변수를 사용하는 정적 SQL문을 표시합니다.

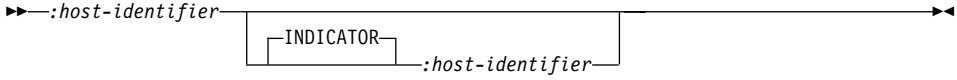
```
INSERT INTO DEPARTMENT
   VALUES (:hv_deptno, :hv_deptname, :hv_mgrno, :hv_admrdept)
```

이 예는 매개변수 표시문자를 사용한 동적 SQL문을 표시합니다.

```
INSERT INTO DEPARTMENT VALUES(?, ?, ?, ?)
```

매개변수 표시문자에 대한 정보는 1087 페이지의 『PREPARE』에 있는 『매개변수 표시문자』에서 참조하십시오.

구문 도표에 있는 메타-변수인 *호스트-변수*는 일반적으로 다음으로 확장될 수 있습니다.



각 *호스트-식별자*는 원시 프로그램에서 선언되어야 합니다. 두 번째 *호스트-식별자*로 지정된 변수의 데이터 유형은 작은 정수여야 합니다.

첫번째 *호스트-식별자*는 주 *변수*를 지정합니다. 연산에 따라, 데이터베이스 관리 프로그램에 값을 제공하거나 데이터베이스 관리 프로그램에서 값을 제공받습니다. 입

력 호스트 변수는 런타임 응용프로그램 코드 페이지에서 값을 제공합니다. 출력 호스트 변수에는 필요한 경우, 데이터가 출력 응용프로그램 변수로 복사될 때 런타임 응용프로그램 코드 페이지로 변환되는 값이 제공됩니다. 주어진 호스트 변수는 같은 프로그램에서 입력과 출력 변수 모두로 제공될 수 있습니다.

두 번째 호스트-식별자는 *표시기 변수*를 지정합니다. 표시기 변수의 목적은 다음을 수행하는 것입니다.

- 널(NULL)값을 지정. 표시기 변수의 음수 값은 널(NULL)값을 지정합니다. 값 -2는 결과를 도출할 때 발생한 산술식 오류나 숫자 변환을 나타냅니다.
- 절단된 문자열의 원래 길이 기록(값의 소스가 대형 오브젝트(LOB) 유형인 경우).
- 시간이 호스트-변수로의 지정에서 절단된 경우 시간의 초 부분을 기록합니다.

예를 들어, 삽입 또는 갱신 값을 지정하기 위해 :HV1:HV2가 사용될 경우, 그리고 HV2가 음수일 경우, 지정된 값은 널(NULL) 값입니다. HV2가 음수가 아닌 경우, 지정되는 값은 HV1값입니다.

마찬가지로, :HV1:HV2가 VALUES INTO 절이나 FETCH 또는 SELECT INTO 문에 지정될 경우, 그리고 리턴되는 값이 널(NULL)일 경우, HV1은 변경되지 않고 HV2는 음수 값으로 설정됩니다. [26] 리턴된 값이 널(NULL)이 아닌 경우, 이 값은 HV1로 지정되고 HV2는 0으로 설정됩니다(이는 HV1로 지정하는 데 비LOB 문자열의 문자열 절단이 요구되지 않는 경우로, 비LOB 문자열의 절단이 요구되는 때에는 HV2가 문자열의 원래 길이로 설정됩니다.). 지정할 때 시간의 두 번째 부분이 절단되어야 하는 경우, HV2는 초 수로 설정됩니다.

두 번째 호스트 식별자가 누락된 경우, 호스트 변수에는 표시기 변수가 없습니다. 호스트 변수 참조 :HV1에 의해 지정되는 값은 항상 HV1의 값이므로, 그 변수에 널(NULL) 값을 지정할 수 없습니다. 그러므로, 이 형식은 해당 컬럼에 널(NULL)

26. 데이터베이스가 DFT_SQLMATHWARN yes를 사용하여 구성된 경우(정적 SQL문의 바인딩 중 구성된 경우), HV2는 -2가 될 수 있습니다. HV2가 -2일 경우, 숫자 유형 HV1로의 변환 오류나 HV1에 대한 값을 결정하는 데 사용된 산술 표현식의 평가시 오류로 인해 HV1에 대한 값이 복귀되지 않을 수 있습니다. DB2 Universal Database 버전 5 이전의 클라이언트 버전을 사용하여 데이터베이스에 액세스하는 경우, 산술 예외로 HV2는 -1이 됩니다.

값을 포함할 수 있는 한, INTO절에서 사용될 수 없습니다. 이 형식이 사용되고, 컬럼에 널(NULL)이 포함된 경우, 데이터베이스 관리 프로그램은 런타임 오류를 발생시킵니다.

호스트 변수를 참조하는 SQL문은 이러한 호스트 변수 선언 범위 내에 있어야 합니다. 커서의 SELECT문에서 참조되는 호스트 변수의 경우, 이 규칙은 DECLARE CURSOR문이 아닌 OPEN문으로 적용됩니다.

예

PROJECT 테이블을 사용하여, 호스트 변수 PNAME(VARCHAR(26))을 프로젝트 이름(PROJNAME)으로 설정하고 호스트 변수 STAFF(dec(5,2))를 중간 스테핑 레벨(PRSTAFF)로 그리고 호스트 변수 MAJPROJ(char(6))를 프로젝트(PROJNO) 'IF1000'의 주 프로젝트(MAJPROJ)로 설정합니다. 컬럼 PRSTAFF 및 MAJPROJ에는 널(NULL) 값이 포함될 수도 있으므로, 표시기 변수 STAFF_IND(smallint)와 MAJPROJ_IND(smallint)를 제공하십시오.

```
SELECT PROJNAME, PRSTAFF, MAJPROJ
  INTO :PNAME, :STAFF :STAFF_IND, :MAJPROJ :MAJPROJ_IND
  FROM PROJECT
  WHERE PROJNO = 'IF1000'
```

MBCS 관련 고려사항: 호스트 변수 이름에 다중 바이트 문자를 사용 여부는 호스트 언어에 의해 결정됩니다.

BLOB, CLOB 및 DBCLOB 호스트 변수 참조

일반적인 BLOB, CLOB 및 DBCLOB 변수, LOB 위치 지정자 변수(160 페이지의 『위치 지정자 변수 참조』 참조) 및 LOB 파일 참조 변수(161 페이지의 『BLOB, CLOB 및 DBCLOB 파일 참조 변수 참조』 참조)는 모든 호스트 언어에서 정의될 수 있습니다. LOB가 허용되는 경우, 구문 도표에서 *호스트 변수*라는 용어는 일반 호스트 변수, 위치 지정자 변수 또는 파일 참조 변수를 의미할 수 있습니다. 이것은 원시 데이터 유형이 아니므로, SQL 확장이 사용되고, 사전 처리 컴파일러는 각 변수를 표현하는 데 필요한 호스트 언어 구조를 생성합니다. REXX의 경우, LOB는 문자열에 대응됩니다.

때때로 대형 오브젝트(LOB) 값 전체를 보관하기에 충분히 큰 변수를 정의할 수도 있습니다. 이것이 가능하고 서버에서 데이터의 전송을 지연시킴으로써 얻게될 성능

상의 이익이 없는 경우, 위치 지정자(locator)는 필요없습니다. 그러나, 호스트 언어나 공간 제한사항이 임시 저장영역에 전체 대형 오브젝트(LOB)를 한번에 보관하는 것에 대해 설명하거나 성능상의 이익이 있으므로, 대형 오브젝트는 한번에 대형 오브젝트(LOB)의 일부만을 포함하는 호스트 변수에서 갱신하거나 호스트 변수에서 선택된 오브젝트의 위치 지정자와 위치를 통해 대형 오브젝트(LOB)를 참조할 수 있습니다.

모든 기타 호스트 변수와 함께, 대형 오브젝트(LOB) 위치 지정자 변수에는 관련 표시기 변수가 들어 있습니다. 대형 오브젝트(LOB) 위치 지정자 호스트 변수에 대한 표시기 변수는 다른 데이터 유형에 대한 표시기 변수와 같은 방법으로 활동합니다. 널(NULL) 값이 데이터베이스에서 리턴되면, 표시기 변수가 설정되고 위치 지정자 호스트 변수는 변경되지 않습니다. 이것은 위치 지정자가 결코 널(NULL) 값으로 지정되지 않음을 의미합니다.

위치 지정자 변수 참조

*위치 지정자 변수*는 응용프로그램 서버(AS)에서의 LOB 값을 나타내는 위치 지정자를 포함하는 호스트 변수입니다. 위치 지정자가 LOB값 조작에 사용되는 방법에 대해서는 89 페이지의 『위치 지정자가 있는 대형 오브젝트(LOB) 조작』에서 참조하십시오.

SQL문에 있는 위치 지정자 변수는 위치 지정자 변수를 선언하는 규칙에 따라 프로그램에서 설명한 위치 지정자 변수를 식별해야 합니다. 이것은 언제나 SQL문을 통해 수행됩니다.

구문 도표에서 사용된 것처럼 위치 지정자란 용어는 위치 지정자 변수 참조를 표시합니다. 메타-변수 *위치 지정자-변수*는 *호스트-변수*에 대한 것과 같은 *호스트 식별자*를 갖도록 확장될 수 있습니다.

위치 지정자와 연관된 표시기 변수가 널(NULL)인 경우, 참조된 LOB의 값은 널(NULL)입니다.

현재 값을 나타내지 않는 위치 지정자-변수가 참조되는 경우, 오류가 발생합니다 (SQLSTATE 0F001).

트랜잭션 확약 또는 모든 트랜잭션 종료시에 이 트랜잭션에서 확보한 모든 위치 지정자가 해제됩니다.

BLOB, CLOB 및 DBCLOB 파일 참조 변수 참조

BLOB, CLOB 및 DBCLOB 파일 참조 변수는 LOB에 대한 직접 파일 입력 및 출력에 사용되고, 모든 호스트 언어에서 정의될 수 있습니다. 이것은 원시 데이터 유형이 아니므로, SQL 확장이 사용되고, 사전 처리 컴파일러는 각 변수를 표현하는 데 필요한 호스트 언어 구조를 생성합니다. REXX의 경우, LOB는 문자열에 대응됩니다.

파일 참조 변수는 LOB 위치 지정자가 LOB 바이트를 포함하지 않고 표현하는 것처럼 파일을 나타냅니다(포함하지 않습니다.). 데이터베이스 조회, 갱신 및 삽입은 하나의 컬럼 값을 보관하거나 검색하는 데 파일 참조 변수를 사용합니다.

파일 참조 변수에는 다음 등록 정보가 있습니다.

데이터 유형
BLOB, CLOB 또는 DBCLOB. 이 등록 정보는 변수가 선언될 때 지정됩니다.

방향
이것은 런타임에 (파일 옵션 값의 부분으로) 응용프로그램에서 지정해야 합니다. 방향은 다음 중 하나입니다.

- 입력(EXECUTE문, OPEN문, UPDATE문, INSERT문 또는 DELETE문에서 데이터의 소스로 사용됨)
- 출력(FETCH문이나 SELECT INTO문에서 데이터의 목적표로 사용됨)

파일 이름
런타임에 응용프로그램에 의해 지정되어야 합니다. 다음 중 하나입니다.

- 파일의 완전 경로 이름(권장사항)
- 상대 파일 이름. 상대 파일 이름이 제공되는 경우, 이것은 클라이언트 프로세스의 현 경로에 첨부됩니다.

응용프로그램 내에서, 한 개의 파일 참조 변수에서 한 개의 파일이 참조되어야 합니다.

파일 이름 길이

런타임에 응용프로그램에 의해 지정되어야 합니다. 파일 이름의 길이입니다(바이트로 표시).

파일 옵션

응용프로그램은 많은 옵션 중 하나를 그 변수를 사용하기 전에 파일 참조 변수에 지정해야 합니다. 옵션은 파일 참조 변수 구조 내의 필드에서 INTEGER 값으로 설정됩니다. 다음 값 중 하나는 각 파일 참조 변수에 대해 지정되어야 합니다.

- 입력(클라이언트▶서버)

  SQL_FILE_READ [27]

  이것은 열리고, 읽히고 또 닫힐 수 있는 일반 파일입니다.

- 출력(서버▶클라이언트로)

  SQL_FILE_CREATE [28]

  새 파일을 작성합니다. 파일이 이미 존재하는 경우, 오류가 발생합니다.

  SQL_FILE_OVERWRITE (Overwrite) [29]

  지정된 이름과 동일한 이름을 갖는 기존 파일이 존재하는 경우 이 파일은 대체되고, 파일이 없으면 새 파일이 작성됩니다.

27. COBOL에서는 SQL-FILE-READ, FORTRAN에서는 sql_file_read, REXX에서는 READ.

28. COBOL에서는 SQL-FILE-CREATE, FORTRAN에서는 sql_file_create, REXX에서는 CREATE.

29. COBOL에서는 SQL-FILE-OVERWRITE, FORTRAN에서는 sql_file_overwrite, REXX에서는 OVERWRITE.

SQL_FILE_APPEND [30]

지정된 이름과 동일한 이름을 갖는 기존 파일이 존재하는 경우 출력은 첨부되고, 파일이 없으면 새 파일이 작성됩니다.

데이터 길이

출력에서는 사용되지 않습니다. 출력에서, 구현은 데이터 길이를 파일에 기록된 새로운 데이터의 길이로 설정합니다. 길이는 바이트 단위입니다.

모든 기타 호스트 변수와 함께, 파일 참조 변수에는 관련 표시기 변수가 들어 있습니다.

출력 파일 참조 변수 예(C로)

- 주어진 선언 절이 다음과 같이 코드화되었습니다.

```
EXEC SQL BEGIN DECLARE SECTION
   SQL TYPE IS CLOB_FILE  hv_text_file;
   char  hv_patent_title[64];
EXEC SQL  END DECLARE SECTION
```

그런 다음, 다음과 같이 입력합니다.

```
EXEC SQL BEGIN DECLARE SECTION
   /* SQL TYPE IS CLOB_FILE  hv_text_file; */
   struct {
       unsigned long  name_length; //  File Name Length
       unsigned long  data_length; //  Data Length
       unsigned long  file_options; // File Options
       char           name[255];   // File Name
   } hv_text_file;
   char  hv_patent_title[64];
EXEC SQL END DECLARE SECTION
```

30. COBOL에서는 SQL-FILE-APPEND, FORTRAN에서는 sql_file_append, REXX에서는 APPEND.

그러면, 다음 코드를 사용하여 데이터베이스의 CLOB 컬럼에서 선택하여 :hv_text_file에 의해 참조되는 새 파일에 넣을 수 있습니다.

```
strcpy(hv_text_file.name, "/u/gainer/papers/sigmod.94");
hv_text_file.name_length = strlen("/u/gainer/papers/sigmod.94");
hv_text_file.file_options = SQL_FILE_CREATE;
EXEC SQL SELECT content INTO :hv_text_file from papers
     WHERE TITLE = 'The Relational Theory behind Juggling';
```

입력 파일 참조 변수 예(C로)

- 위와 같은 선언 섹션을 사용할 경우, 다음 코드를 사용하여 :hv_text_file에 의해 참조되는 일반 파일에서 CLOB 컬럼으로 데이터를 삽입할 수 있습니다.

```
strcpy(hv_text_file.name, "/u/gainer/patents/chips.13");
hv_text_file.name_length = strlen("/u/gainer/patents/chips.13");
hv_text_file.file_options = SQL_FILE_READ:
strcpy(:hv_patent_title, "A Method for Pipelining Chip Consumption");
EXEC SQL INSERT INTO patents( title, text )
         VALUES(:hv_patent_title, :hv_text_file);
```

구조화 유형 호스트 변수에 대한 참조

구조화 유형 변수는 FORTRAN, REXX 및 Java를 제외한 모든 호스트 언어로 정의될 수 있습니다. 이것은 원시 데이터 유형이 아니므로, SQL 확장이 사용되고, 사전 처리 컴파일러는 각 변수를 표현하는 데 필요한 호스트 언어 구조를 생성합니다.

모든 기타 호스트 변수와 함께, 구조화 유형 변수에는 관련 표시기 변수가 들어 있습니다. 구조화 유형 호스트 변수에 대한 표시기 변수는 다른 데이터 유형에 대한 표시기 변수와 같은 방법으로 활동합니다. 널(NULL) 값이 데이터베이스에서 리턴되면, 표시기 변수가 설정되고 구조화 유형 호스트 변수는 변경되지 않습니다.

구조화 유형에 대한 실제 호스트 변수는 내장 데이터 유형으로 정의됩니다. 구조화 유형과 연관되는 내장 데이터 유형은 다음과 같이 지정할 수 있어야 합니다.

- 사전 처리 컴파일 명령에서 지정된 TRANSFORM GROUP 옵션에 의해 정의된 구조화 유형에 대한 FROM SQL 변환 함수의 결과로부터, 그리고
- 사전 처리 컴파일 명령에서 지정된 TRANSFORM GROUP 옵션에 의해 정의된 구조화 유형에 대한 TO SQL 변환 함수의 매개변수로.

호스트 변수 대신 매개변수 표시문자를 사용할 경우, 적절한 매개변수 유형 특성은 SQLDA에 지정해야 합니다. 이 때, SQLDA에서 SQLVAR 구조의 "doubled" 세트와, 2차 SQLVAR의 SQLDATATYPE_NAME 필드는 구조화 유형의 스키마와 유형 이름으로 채워져야 합니다. SQLDA 구조에서 스키마가 생략될 경우, 오류가 발생합니다(SQLSTATE 07002). 이 주제에 대해서는 1267 페이지의 『부록 C. SQL 설명자 영역(SQLDA)』에서 자세한 내용을 참조하십시오.

예

C 프로그램에서 내장 유형 BLOB(1048576)를 사용하여, 유형 POLYGON의 호스트 변수 *hv_poly* 및 *hv_point*를 정의하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
     static SQL
        TYPE IS POLYGON AS BLOB(1M)
        hv_poly, hv_point;
EXEC SQL END DECLARE SECTION;
```

함수

*데이터베이스 함수*는 입력 데이터 값 집합과 결과 값 집합사이의 관계입니다. 예를 들어, TIMESTAMP 함수는 데이터 유형이 DATE와 TIME인 입력 데이터 값을 입력하고, 결과는 TIMESTAMP입니다. 함수는 내장되었거나 사용자가 정의할 수 있습니다.

- *내장 함수*는 데이터베이스 관리 프로그램에 의해 제공되어 하나의 결과 값을 얻고, SYSIBM 스키마의 일부로 식별됩니다. 그러한 함수의 예로는 AVG와 같은 컬럼 함수, "+"와 같은 연산자 함수, 십진수와 같은 변환 함수 및 SUBSTR과 같은 기타 함수가 있습니다.
- *사용자 정의 함수*는 SYSCAT.FUNCTIONS(CREATE FUNCTION문을 사용)에 있는 데이터베이스에 등록된 함수입니다. 사용자 정의 함수는 SYSIBM 스키마의 일부가 아닙니다. 그러한 함수 집합 중 하나가 SYSFUN 스키마에서 데이터베이스 관리 프로그램과 함께 제공됩니다.

사용자 정의 함수를 사용하면, DB2는 사용자와 응용프로그램 개발자가 사용자나 제3의 공급자가 데이터베이스 엔진 자체에 적용하도록 제공한 함수 정의를 추가하여 데이터베이스 시스템의 기능을 확대할 수 있습니다. 이렇게 하면 데이터베이스에서 행을 검색하여 검색된 데이터에 이들 함수를 적용하는 것보다 더 나은 성능

이 얻어지므로, 데이터 감축을 더욱 제한하거나 데이터 감축을 수행할 수 있습니다. 데이터베이스 기능을 확장함으로써 데이터베이스는 응용프로그램이 사용하는 엔진에서 동일한 기능을 발휘하며, 응용프로그램과 데이터베이스 사이에서 더 많은 상승효과를 가져오며, 더욱 오브젝트 지향적이므로 응용프로그램 개발자의 생산성 향상에 기여합니다.

SYSIBM 및 SYSFUN 스키마의 전체 함수 목록은 240 페이지의 표15에 나와 있습니다.

외부, SQL 및 전래 사용자 정의 함수

사용자 정의 함수는 *외부 함수*, *SQL 함수* 또는 *전래 함수*일 수 있습니다. *외부 함수*는 오브젝트 코드 라이브러리를 참조하는 데이터베이스와 함수가 호출되면 실행될 그 라이브러리 내의 함수로 정의됩니다. 외부 함수는 컬럼 함수가 아닙니다. *전래(sourced) 함수*는 데이터베이스에 이미 알려진 다른 내장 함수나 사용자 정의 함수를 참조하는 데이터베이스에 정의됩니다. 전래(sourced) 함수는 스칼라 함수 또는 컬럼 함수가 될 수 있습니다. 이들은 사용자 정의 유형을 사용하여 기존 함수의 사용을 지원할 경우 매우 유용합니다. *SQL 함수*는 SQL RETURN문만 사용하여 데이터베이스에 대해 정의됩니다. 이 함수는 스칼라 값, 행 또는 테이블을 리턴할 수 있습니다. SQL 함수는 컬럼 함수가 될 수 없습니다.

스칼라, 컬럼, 행 및 테이블 사용자 정의 함수

각 사용자 정의 함수는 *스칼라*, *컬럼* 또는 *테이블* 함수로 분류되기도 합니다.

*스칼라 함수*는 호출될 때마다 단일 값의 응답을 복귀시키는 함수입니다. 예를 들어, 내장 함수 SUBSTR()은 스칼라 함수입니다. 스칼라 UDF는 외부 또는 전래 함수일 수 있습니다.

*컬럼 함수*는 개념적으로 유사 값 세트(한 컬럼)로 전달되고 단일 값으로 응답을 리턴하는 함수입니다. 이 함수는 DB2에서 *총계 함수*라고도 합니다. 컬럼 함수의 예로는 내장 함수 AVG()가 있습니다. 외부 컬럼 UDF는 DB2로 정의할 수 없지만 내장 컬럼 함수 중 하나에 근거하여 발생된 컬럼 UDF는 정의할 수 있습니다. 이는 구별 유형에 유용합니다. 예를 들어, 기본 유형 INTEGER와 더불어 구별 유형 SHOESIZE가 정의된 경우, 내장 함수 AVG(ITNEGER)에 근거하여 발생된 UDF AVG(SHOESIZE)를 정의할 수 있으며 이것이 컬럼 함수가 됩니다.

*행 함수*는 한 행 값을 리턴하는 함수입니다. 이 함수는 구조화 유형의 속성 값을 한 행의 값들에 맵핑하여 변환 함수로만 사용할 수 있습니다. 행 함수는 SQL 함수로 정의될 수 있습니다.

*테이블 함수*는 테이블을 참조하는 SQL문에 리턴하는 함수입니다. 테이블 함수는 SELECT문의 FROM절에서만 참조할 수 있습니다. 이러한 함수는 SQL 언어 처리 능력을 DB2 데이터가 아닌 데이터에 적용하거나, 이러한 데이터를 DB2 테이블로 변환하는 데 사용할 수 있습니다. 예를 들어, 파일을 취하여 이를 테이블로 변환하고, WWW에서 샘플 데이터를 취하여 이를 표로 만들 수 있거나, Lotus Notes 데이터베이스에 액세스하여 데이터, 보낸 사람 및 메시지 텍스트와 같이 메일 메시지에 대한 정보를 리턴합니다. 이 정보를 데이터베이스의 다른 테이블과 조인할 수 있습니다. 테이블 함수는 외부 함수나 SQL 함수로 정의될 수 있으며, 전래 함수가 될 수는 없습니다.

함수 시그니처

함수는 스키마, 함수 이름, 매개변수의 수 및 그 매개변수의 데이터 유형으로 식별됩니다. 이것은 데이터베이스 내에서 고유해야 하며, *함수 시그니처*라고 합니다. 매개변수의 수나 매개변수의 데이터 유형이 다르게 제공되면, 스키마에 같은 이름의 함수가 둘 이상 있을 수 있습니다. 여러 함수 인스턴스가 있는 함수 이름을 *오버로드(overload)* 함수라고 합니다. 함수 이름은 스키마 내에서 오버로드될 수 있으며, 이런 경우 스키마에서 그 이름으로 둘 이상의 함수가 있을 수 있습니다(필요한 경우 매개변수 유형은 다릅니다.). 함수 이름은 SQL 경로에서 오버로드될 수 있으며, 이런 경우 경로에 그 이름으로 하나 이상의 함수가 있으며, 이러한 함수는 다른 매개변수 유형을 가질 필요가 없습니다.

SQL 경로

함수는 괄호로 묶인 인수 목록이 뒤에 오는 규정된 이름(스키마 및 함수 이름)을 허용되는 문맥으로 참조하여 호출할 수 있습니다. 함수는 또한 스키마 이름 없이 호출되어 동일한 매개변수나 허용가능한 매개변수를 가진 다른 스키마를 선택할 수 있습니다. 이런 경우, *SQL 경로*는 *함수 차수*를 지원하는 데 사용됩니다. SQL 경로는 이름, 매개변수 개수 및 승인 가능한 데이터 유형이 같은 함수를 식별하기 위해 검색되는 스키마 목록입니다. 정적 SQL문의 경우, SQL 경로는 FUNCPATH

바인드 옵션을 사용하여 지정됩니다(*Command Reference* 참조). 동적 SQL문의 경우, SQL 경로는 CURRENT PATH 특수 레지스터의 값입니다(142 페이지의 『CURRENT PATH』 참조).

함수 차수

함수가 호출되면, 데이터베이스 관리 프로그램은 같은 이름을 가진 가능한 함수 중 어느것이 『최적』인지 결정해야 합니다. 이는 내장 함수와 사용자 정의 함수에서의 함수 해석이 포함됩니다.

*인수(argument)*는 호출시 함수에 전달되는 값입니다. 함수가 SQL에서 호출되면, 이것은 하나 이상의 인수 목록에 전달됩니다. 인수의 의미는 인수 목록의 위치에 의해 결정됩니다. *매개변수*는 함수 입력의 공식적인 정의입니다. 함수가 데이터베이스로 정의될 때, 내부적으로(내장 함수) 또는 사용자(사용자 정의 함수)에 의해 매개변수(0개 또는 그 이상)가 지정되고, 이들의 의미와 위치를 정의하는 정의 순서도 지정됩니다. 그러므로, 모든 매개변수는 함수의 특정 위치 입력입니다. 호출시, 인수(argument)는 인수 목록에 있는 위치에 의해 특정 매개변수에 대응됩니다.

데이터베이스 관리 프로그램은 호출시 제공된 함수 이름, 인수의 개수와 데이터 유형, SQL 경로에서 같은 이름을 가진 모든 함수 및 해당 매개변수의 데이터 유형을 함수의 선택 여부를 결정하기 위한 기준으로서 사용합니다. 다음은 결정 프로세스의 가능한 결과입니다.

1. 특정 함수가 최적으로 판단됩니다. 예를 들어, 스키마 TEST에 다음과 같이 함수 경로가 정의된 시그니처를 갖는 함수 이름 RISK가 있습니다.

   ```
   TEST.RISK(INTEGER)
   TEST.RISK(DOUBLE)
   ```

   SQL 경로에는 TEST 스키마 및 다음과 같은 함수 참조(여기서, DB는 DOUBLE 컬럼)가 포함됩니다.

   ```
   SELECT ... RISK(DB) ...
   ```

   그러면, 두 번째 RISK가 선택됩니다.

   다음 함수 참조(여기서, SI은 SMALLINT 컬럼)는

```
SELECT ... RISK(SI) ...
```

첫번째 RISK를 선택하는 데, 이는 SMALLINT가 INTEGER로 승격될 수 있고, 선행 목록 아래에 있는 DOUBLE보다 더 적합하기 때문입니다(104 페이지의 표5에 표시된 대로).

구조화 유형인 인수를 고려할 때, 앞의 목록에 인수의 정적 유형에 대한 수퍼 유형이 포함됩니다. 최적의 함수는 구조화 유형 계층에서 함수 인수의 정적 유형에 가장 가까이에 있는 수퍼 유형 매개변수로 정의된 함수입니다.

2. 적합하도록 선택된 함수가 없습니다. 예를 들어, 이전 예에서와 같은 두 함수와 다음의 함수 참조가 있습니다(여기서, C는 CHAR(5) 컬럼).

   ```
   SELECT ... RISK(C) ...
   ```

   인수는 어느 RISK 함수의 매개변수와도 일치하지 않습니다.

3. 특정 경로는 SQL 경로 및 호출시 전달된 인수의 갯수와 데이터 유형을 기준으로 선택됩니다. 예를 들어, 함수 이름 RANDOM이 다음과 같이 정의되었다고 합시다.

   ```
   TEST.RANDOM(INTEGER)
   PROD.RANDOM(INTEGER)
   ```

   또한, SQL 경로는 다음과 같습니다.

   ```
   "TEST","PROD"
   ```

   다음의 함수 참조는

   ```
   SELECT ... RANDOM(432) ...
   ```

   TEST.RANDOM을 선택합니다. 이는 RANDOM 함수가 거의 모두 일치(특수한 경우에는 정확히 일치함)하고 두 스키마가 모두 경로 내에 있기는 하지만, TEST가 SQL 경로에서 PROD 앞에 오기 때문입니다.

최적(Best fit) 선택 방법

고려중인 함수 매개변수의 정의된 데이터 유형과 인수의 데이터 유형의 비교는 비슷한 이름의 함수 그룹에서 어느 함수가 "최적"인지를 결정하는 기준입니다. 함수의 결과 데이터 유형 또는 고려중인 함수의 유형(컬럼, 스칼라 또는 테이블)은 이 결정에 영향을 주지 **않습니다**.

함수 차수는 다음 단계를 사용하여 수행됩니다.

1. 먼저, 카탈로그(SYSCAT.FUNCTIONS) 및 내장 함수에서 다음이 모든 함수들을 찾습니다.
   a. *스키마* 이름이 지정된(즉, *규정화된* 참조) 호출의 경우, *스키마* 이름과 함수 이름이 호출 이름과 일치합니다.
   b. *스키마* 이름이 지정되지 않은(즉, *규정되지 않은* 참조) 호출의 경우, 함수 이름이 호출 이름과 일치하고 *스키마* 이름이 SQL 경로에 있는 *스키마* 중 하나와 일치합니다.
   c. 정의된 매개변수 개수가 호출과 일치합니다.
   d. 각 호출 인수는 데이터 유형에서 함수에 대해 해당되는 정의된 매개변수와 일치하거나, 그 매개변수로 『승격할 수 있습니다』(104 페이지의 『데이터 유형의 승격』 참조).
2. 다음으로, 왼쪽에서 오른쪽으로 함수 호출의 각 인수를 검사합니다. 각각의 인수에 대해, 그 인수에 *최적*이 아닌 모든 함수를 제거합니다. 주어진 인수에 대해 *최적*은 104 페이지의 표5에 있는 인수 데이터 유형에 해당하는 순서 목록에서 처음 나타나는 데이터 유형으로, 여기에는 그 데이터 유형의 매개변수를 가진 함수가 있습니다. 길이, 정밀도, 스케일 및 ″FOR BIT DATA″ 속성은 이 비교에서 고려되지 않습니다. 예를 들어, 십진수(9,1) 인수는 십진수(6,5) 매개변수와 정확히 일치하는 것으로 고려되고, VARCHAR(19) 인수는 VARCHAR(6) 매개변수와 정확히 일치하는 것으로 고려됩니다.

   사용자가 정의하는 구조화 유형 인수에 대해 최적의 일치는 자체입니다. 다음 최적 일치는 해당되는 중간 수퍼 유형이며, 인수의 각 수퍼 유형에 대해 마찬가지로 진행됩니다. 구조화 유형 인수의 유일한 정적 유형(선언된 유형)이 고려되고, 동적 유형(대부분의 특정 유형)은 고려되지 않습니다.
3. 2단계 이후에 둘 이상의 후보 함수가 있는 경우, 나머지 후보 함수에 동일한 시그니처가 있지만 다른 *스키마*에 있는 경우(알고리즘이 동작하는 방법)가 되어야 합니다. 사용자의 SQL 경로에서 스키마가 가장 빠른 함수를 선택하십시오.
4. 2단계 이후에 남아 있는 후보 함수가 없는 경우, 오류가 발생합니다 (SQLSTATE 42884).

내장 함수에 대한 함수 경로 고려사항

내장 함수는 SYSIBM이라는 특수 스키마에 있습니다. 그 밖의 함수는 내장 함수에서 고려되지 않는 SYSFUN 스키마에서 사용할 수 있습니다. 이는 이것이 사용자 정의 함수로서 개발되었고 특별한 처리 고려사항이 없기 때문입니다. 사용자는 SYSIBM 또는 SYSFUN 스키마에서(아니면, 『SYS』로 시작하는 이름의 스키마에서) 추가 함수를 정의할 수 없습니다.

이미 언급한 바와 같이, 내장 함수는 이들이 사용자 정의 함수에서 수행한 것과 같이 함수 차수 프로세스에도 참여합니다. 함수 차수 측면에서의 내장 함수와 사용자 정의 함수의 한 가지 차이점은 내장 함수는 언제나 함수 차수로 고려되어야 한다는 것입니다. 그러므로, 경로에서 SYSIBM을 생략하면 함수와 데이터 유형 해석에 대해 SYSIBM이 경로상에서 첫번째 스키마로 간주되게 됩니다.

예를 들어, 사용자의 SQL 경로가 다음과 같이 정의되고,

```
"SHAREFUN","SYSIBM","SYSFUN"
```

SYSIBM.LENGTH와 인수 개수 및 유형이 같은 LENGTH 함수가 스키마 SHAREFUN에 정의된 경우, 이 사용자의 SQL문에서 LENGTH에 대해 규정되지 않은 참조를 할 경우 SHAREFUN.LENGTH가 선택됩니다. 그러나, 사용자의 SQL 경로가 다음과 같이 정의되고

```
"SHAREFUN","SYSFUN"
```

동일한 SHAREFUN.LENGTH 함수가 존재할 경우, 이 사용자의 SQL문에서 LENGTH에 대해 규정되지 않은 참조를 할 경우 SYSIBM.LENGTH가 선택됩니다. SYSIBM을 지정하지 않아서 이것이 경로에서 내재적으로는 첫번째가 되기 때문입니다.

다음을 수행하여 이 영역에서 잠재적인 문제점을 최소화할 수 있습니다.

- 사용자 정의 함수에 대해 내장 함수 이름을 절대 사용하지 않거나
- 내장 함수와 같은 이름으로 사용자 정의 함수를 작성할 필요가 있다고 생각된 경우, 이러한 함수의 모든 참조를 규정화합니다.

함수 차수 예

다음은 성공적인 함수 차수의 예입니다.

세 개의 서로 다른 스키마에서 7개의 FOO 함수가 등록되어 있습니다. 모든 필수 키워드를 표시한 것은 아니라는 점에 유의하십시오.

```
CREATE FUNCTION AUGUSTUS.FOO(CHAR(5), INT, DOUBLE)  SPECIFIC FOO_1 ...
CREATE FUNCTION AUGUSTUS.FOO(INT, INT, DOUBLE)      SPECIFIC FOO_2 ...
CREATE FUNCTION AUGUSTUS.FOO(INT, INT, DOUBLE, INT) SPECIFIC FOO_3 ...
CREATE FUNCTION JULIUS.FOO  (INT, DOUBLE, DOUBLE)   SPECIFIC FOO_4 ...
CREATE FUNCTION JULIUS.FOO  (INT, INT, DOUBLE)      SPECIFIC FOO_5 ...
CREATE FUNCTION JULIUS.FOO  (SMALLINT, INT, DOUBLE) SPECIFIC FOO_6 ...
CREATE FUNCTION NERO.FOO    (INT, INT, DEC(7,2))    SPECIFIC FOO_7 ...
```

함수 참조는 다음과 같습니다.(여기서, I1과 I2는 INTEGER 컬럼이고, D는 십진수 컬럼입니다.)

```
SELECT ... FOO(I1, I2, D) ...
```

이 참조를 구성하는 적용업무에 다음과 같이 설정될 수 있는 SQL 경로가 있다고 가정합니다.

```
"JULIUS","AUGUSTUS","CAESAR"
```

이 알고리즘을 따라 수행하면...

FOO_7은 스키마 "NERO"가 SQL 경로에 포함되지 않으므로, 후보에서 제거됩니다.

FOO_3은 매개변수의 개수가 틀리므로 후보에서 제거됩니다. FOO_1과 FOO_6은 두 경우 모두 첫번째 인수가 첫번째 매개변수의 데이터 유형으로 승격될 수 없으므로 제거됩니다.

하나 이상의 후보가 남아 있으므로, 인수는 순서대로 고려되어야 합니다.

첫번째 인수의 경우, 나머지 함수(FOO_2, FOO_4, FOO_5)는 인수 유형과 정확히 일치합니다. 이 고려사항에서는 아무 것도 삭제되지 않았으므로 다음 인수가 검사되어야 합니다.

두 번째 인수의 경우, FOO_2와 FOO_5는 정확히 일치하지만, FOO_4는 일치하지 않으므로, 고려 대상에서 제외됩니다. 다음 인수는 FOO_2와 FOO_5의 차이점을 결정하기 위해 사용됩니다.

세번째와 마지막 인수에서, FOO_2나 FOO_5 모두 인수 유형과 정확히 일치하지는 않지만 비교적 양호합니다.

남아있는 두 함수 FOO_2와 FOO_5에는 똑같은 매개변수 시그니처가 있습니다. 최종 타이-브레이커(tie-breaker)는 SQL 경로에서 처음으로 제공되는 함수 스키마를 찾는 것입니다. 이 기준에서, FOO_5가 마지막으로 선택됩니다.

함수 호출

일단 함수가 선택되어도, 함수를 사용할 수 없는 이유가 여전히 있습니다. 각 함수는 특정 데이터 유형을 가진 결과를 리턴하도록 정의됩니다. 이 결과 데이터 유형이 함수가 호출된 문맥과 호환되지 않을 경우, 오류가 발생합니다. 예를 들어, STEP 함수 이름이 정의되고, 이때 다음 결과와 같이 데이터 유형이 다르고,

```
STEP(SMALLINT) returns CHAR(5)
STEP(DOUBLE)   returns INTEGER
```

함수 참조가 다음과 같은 경우(여기서 S는 SMALLINT 컬럼),

```
SELECT ... 3 + STEP(S) ...
```

인수 유형과 정확히 일치하는 것이 있으므로, 첫번째 STEP이 선택됩니다. 결과 유형이 추가 연산자의 인수에 필요한 숫자 유형이 아닌 CHAR(5)이므로 명령문에서 오류가 발생합니다.

이러한 상황이 발생할 수 있는 또다른 예로는 다음이 있으며, 두 경우 모두 명령문에서의 오류로 귀착됩니다.

1. 함수가 FROM절에서 참조되었지만, 함수 차수 단계에서 선택된 함수가 스칼라 함수이거나 컬럼 함수였습니다.
2. 위와 반대되는 경우로, 문맥에서 스칼라 함수나 컬럼 함수를 호출하고, 함수 차수가 테이블 함수를 선택합니다.

함수 호출의 인수가 선택된 함수의 매개변수 데이터 유형과 정확히 일치하지 않는 경우, 인수는 칼럼에 지정과 같은 규칙을 사용하여 수행시 매개변수의 데이터 유형으로 변환됩니다(109 페이지의 『지정 및 비교』 참조). 여기에는 인수와 매개변수의 정밀도, 스케일 또는 길이가 다른 경우와 포함됩니다.

메소드

구조화 유형의 데이터베이스 메소드는 입력 데이터 값 세트와 결과 값 세트 사이의 관계입니다. 여기서 첫번째 입력 값(또는 *주제 인수*)은 유형이 같거나 메소드의 주제 유형에 대한 부속 유형입니다(*주제 매개변수*라고도 함). 예를 들어, 유형이 ADDRESS인 CITY라고 하는 메소드는 유형 VARCHAR의 입력 데이터 값으로 전달될 수 있고 결과는 ADDRESS(또는 ADDRESS의 부속 유형)입니다.

메소드는 사용자가 정의하는 구조화 유형의 정의 일부로 내재적이나 명시적으로 정의됩니다.

내재적으로 정의된 메소드는 구조화된 유형마다 작성됩니다. Observer 메소드는 구조화 유형의 각 속성에 대해 정의됩니다. Observer 메소드는 응용프로그램이 해당 유형의 인스턴스에 대한 속성 값을 갖도록 허용합니다. Mutator 메소드는 또한 각 속성에 대해 정의되어, 응용프로그램이 유형 인스턴스의 속성 값을 변경하여 유형 인스턴스를 변경할 수 있게 합니다. 위에 설명된 CITY 메소드는 유형 ADDRESS에 대한 mutator 메소드의 예입니다.

명시적으로 정의된 메소드나, *사용자 정의 메소드*는 CREATE TYPE(또는 ALTER TYPE ADD METHOD) 및 CREATE METHOD 명령문의 조합을 사용하여 SYSCAT.FUNCTIONS에서 데이터베이스에 등록된 메소드입니다. 구조화 유형에 대해 정의된 모든 메소드는 유형과 같은 스키마에서 정의됩니다.

구조화 유형에 대해 사용자 정의 함수를 사용하면, DB2는 사용자와 응용프로그램 개발자가 데이터베이스 시스템의 함수를 확장할 수 있습니다. 이는 데이터베이스 엔진에서 구조화 유형 인스턴스에 적용되며 사용자나 써드 파티 벤더에서 제공되는 메소드 정의를 추가하여 수행됩니다. 추가된 메소드 정의는 데이터베이스에서 행을 검색하고 검색된 데이터에 대해 함수를 적용하는 것과는 반대로, 높은 수준의 성능을 제공합니다. 데이터베이스 메소드를 정의해도 데이터베이스는 응용프로그램에서 사용되는 엔진에서 동일한 메소드를 노출하여, 응용프로그램과 데이터베이스 사이에 더 나은 수준의 상호 효율성을 제공합니다. 이것은 오브젝트 지향 성격이 더 크므로, 응용프로그램 개발자에게 더 나은 생산성을 제공해 줍니다.

외부 및 SQL 사용자 정의 메소드

사용자 정의 메소드는 외부 메소드이거나 SQL 표현식을 기초로 할 수 있습니다. 외부 메소드는 오브젝트 코드 라이브러리를 참조하는 데이터베이스와 메소드가 호출되면 실행될 그 라이브러리 내의 함수로 정의됩니다. SQL 표현식을 기초로 하는 메소드는 메소드가 호출될 때 SQL 표현식 결과를 리턴합니다. 그러한 메소드는 완전하게 SQL로 작성되므로 오브젝트 코드 라이브러리가 필요하지 않습니다.

사용자 정의 메소드는 호출될 때마다 단일 값의 응답을 리턴할 수 있습니다. 이 값은 구조화 유형이 될 수 있습니다. 메소드는 주제 인수의 동적 유형이 메소드의 리턴된 유형으로 리턴될 수 있도록 *유형 보존*으로 정의할 수 있습니다(SELF AS RESULT를 사용하여). 내재적으로 정의된 모든 mutator 메소드는 유형이 보존됩니다.

메소드 시그니처

메소드는 주제 유형, 메소드 이름, 매개변수의 수 및 그 매개변수의 데이터 유형으로 식별됩니다. 이것을 메소드 시그니처라고 하며, 이는 데이터베이스 내에서 고유해야 합니다.

다음과 같은 경우, 구조화 유형에 대해 같은 이름의 하나 이상의 메소드가 존재할 수 있습니다.

- 매개변수의 수나 매개변수의 데이터 유형이 다를 경우
- 메소드 주제 유형의 부속 유형이나 수퍼 유형에 대해 같은 시그니처가 존재하지 않을 경우
- 같은 함수 시그니처(주제 유형을 사용하거나, 첫번째 매개변수로 부속 유형 또는 수퍼 유형을 사용하는)가 존재하지 않습니다.

다중 메소드 인스턴스를 가지고 있는 메소드 이름을 *오버로드된 메소드*라고 합니다. 메소드 이름은 한 유형 내에서 오버로드될 수 있으며, 이런 경우 유형에 대해 그 이름으로 하나 이상의 메소드가 있을 수 있습니다(모두 매개변수 유형은 다릅니다). 메소드 이름은 주제 유형 계층 구조에서 오버로드될 수 있으며, 이런 경우 유형 계층에 그 이름으로 하나 이상의 메소드가 있으며, 이러한 메소드는 다른 매개변수 유형을 가지고 있어야 합니다.

메소드 호출

메소드는 허용되는 문맥에서, 앞에 구조화 유형 인스턴스(주제 인수)와 이중 점 연산자가 있는 메소드 이름을 창조함으로써 호출할 수 있습니다. 괄호로 묶어 있는 인수 목록은 그대로 따라야 합니다. 실제로 호출되는 메소드는 다음 절에 설명된 메소드 분석을 사용하여, 주제 유형의 정적 유형을 기초로 판별됩니다. WITH FUNCTION ACCESS로 정의된 메소드는 함수 호출로 호출할 수도 있습니다. 이러한 경우, 함수 결정에 대한 일반 규칙이 적용됩니다.

메소드 분석

메소드가 호출되면, 데이터베이스 관리 프로그램은 같은 이름을 가진 가능한 메소드 중 어느 것이 ″최적″인지 결정해야 합니다. 함수(내장 또는 사용자 정의)는 메소드 분석 중에 고려되지 않습니다.

인수는 호출시 메소드에 전달되는 값입니다. 메소드가 SQL에서 호출되면, 이것은 주제 인수(구조화 유형의)와, 0개 이상의 인수 목록으로 전달됩니다. 인수의 의미는 인수 목록의 위치에 의해 결정됩니다. 주제 인수가 첫번째 인수로 간주됩니다. 매개변수는 메소드 입력의 공식적인 정의입니다.

메소드가 데이터베이스에 대해 정의될 때, 내재적으로 유형에 대해(시스템 생성) 또는 사용자(사용자 정의 메소드)에 의해 매개변수가 지정되고(첫번째 매개변수로 주제 매개변수 사용), 해당되는 정의의 순서로 위치와 의미가 정의됩니다. 그러므로, 모든 매개변수는 메소드의 특정 위치 입력입니다. 호출시, 인수는 인수 목록에 있는 위치에 의해 특정 매개변수에 대응됩니다.

데이터베이스 관리 프로그램은 호출시 제공된 메소드 이름, 인수의 개수와 데이터 유형, 주제 인수의 정적 유형(그리고, 해당되는 수퍼 유형)에 대한 동일한 이름의 모든 메소드, 그리고 해당 매개변수의 데이터 유형을 메소드의 선택 여부를 결정하기 위한 기준으로서 사용합니다.

다음은 결정 프로세스의 가능한 결과입니다.

1. 특정 메소드가 최적으로 판단됩니다. 예를 들어, 시그니처가 다음과 같이 정의된 유형 SITE에 대한 RISK 메소드가 있을 경우,

   ```
   PROXIMITY(INTEGER) FOR SITE
   PROXIMITY(DOUBLE) FOR SITE
   ```

다음 메소드 호출(ST는 SITE 컬럼, DB는 DOUBLE 컬럼)은

```
SELECT ST..PROXIMITY(DB) ...
```

두 번째 PROXIMITY가 선택되도록 합니다.

다음 메소드 호출(여기서, SI은 SMALLINT 컬럼)은

```
SELECT ST..PROXIMITY(SI) ...
```

첫번째 PROXIMITY를 선택하는 데, 이는 SMALLINT가 INTEGER로 승격될 수 있고, 선행 목록 아래에 있는 DOUBLE보다 더 적합하기 때문입니다.

구조화 유형인 인수를 고려할 때, 앞의 목록에 인수의 정적 유형에 대한 수퍼 유형이 포함됩니다. 최적의 함수는 구조화 유형 계층에서 함수 인수의 정적 유형에 가장 가까이에 있는 수퍼 유형 매개변수로 정의된 함수입니다.

2. 적합하도록 선택된 메소드가 없습니다. 예를 들어, 이전 예에서와 같은 두 함수와 다음의 함수 참조가 있습니다(여기서, C는 CHAR(5) 컬럼).

   ```
   SELECT ST..PROXIMITY(C) ...
   ```

   인수는 어느 PROXIMITY 함수의 매개변수와도 일치하지 않습니다.

3. 특정 메소드는 유형 계층에서 메소드와 호출시 전달된 인수의 개수 및 데이터 유형을 기준으로 선택됩니다. 예를 들어, 시그니처가 다음과 같이 정의된 유형 SITE 및 DRILLSITE(SITE의 부속 유형)에 대한 RISK 메소드가 있을 경우,

   ```
   RISK(INTEGER) FOR DRILLSITE
   RISK(DOUBLE) FOR SITE
   ```

   다음 메소드 호출(DRST는 DRILLSITE 컬럼, DB는 DOUBLE 컬럼)은

   ```
   SELECT DRST..RISK(DB) ...
   ```

   DRILLSITE가 SITE로 승격될 수 있으므로, 두 번째 RISK가 선택되도록 합니다.

   다음 메소드 참조(여기서, SI은 SMALLINT 컬럼)는

   ```
   SELECT DRST..RISK(SI) ...
   ```

첫번째 RISK를 선택하는 데, 이는 SMALLINT가 DOUBLE보다 선행 목록에서 더 가까이에 있는 INTEGER로 승격될 수 있고, DRILLSITE가 수퍼 유형인 SITE보다 더 적합하기 때문입니다.

같은 유형 계층 내의 메소드는 같은 시그니처를 수반할 수 없습니다(주제 매개변수 이외의 다른 매개변수를 고려합니다).

최적(Best fit) 선택 방법

고려하는 메소드 매개변수의 정의된 데이터 유형과 인수의 데이터 유형을 비교하는 것은 비슷한 이름의 메소드 그룹에서 어느 메소드가 "최적"인지를 결정하는 기준입니다. 고려하는 메소드의 결과 데이터 유형은 이 결정에 영향을 주지 않습니다.

메소드 분석은 다음 단계를 사용하여 수행됩니다.

1. 먼저, 카탈로그(SYSCAT.FUNCTIONS)에서 다음의 모든 사항이 만족되는 모든 메소드를 찾습니다.
   - 메소드 이름이 호출 이름과 일치하고, 주제 매개변수가 같은 유형이거나 주제 인수의 정적 유형에 대한 수퍼 유형입니다.
   - 정의된 매개변수 개수가 호출과 일치합니다.
   - 각 호출 인수가 데이터 유형에서 메소드의 해당되는 정의된 매개변수와 일치하거나, 그 매개변수로 승격할 수 있습니다(104 페이지의 『데이터 유형의 승격』 참조).
2. 다음으로, 왼쪽에서 오른쪽으로 메소드 호출의 각 인수를 검사합니다. 가장 왼쪽에 있는 인수(첫번째 인수)는 내재된 SELF 매개변수입니다. 예를 들어, 유형 ADDRESS_T에 대해 정의된 메소드는 유형 ADDRESS_T의 내재된 첫번째 인수를 수반합니다.

   각각의 인수에 대해, 그 인수에 최적이 아닌 모든 함수를 제거합니다. 주어진 인수에 대해 최적은 104 페이지의 표5에 있는 인수 데이터 유형에 해당하는 순서 목록에서 처음 나타나는 데이터 유형으로, 여기에는 그 데이터 유형의 매개변수를 가지고 있는 함수가 있습니다.

길이, 정밀도, 스케일 및 ″FOR BIT DATA″ 속성은 이 비교에서 고려되지 않습니다. 예를 들어, 십진수(9,1) 인수는 십진수(6,5) 매개변수와 정확히 일치하는 것으로 고려되고, VARCHAR(19) 인수는 VARCHAR(6) 매개변수와 정확히 일치하는 것으로 고려됩니다.

사용자가 정의하는 구조화 유형 인수에 대해 최적의 일치는 자체입니다. 다음 최적 일치는 해당되는 중간 수퍼 유형이며, 인수의 각 수퍼 유형에 대해 마찬가지로 진행됩니다. 구조화 유형 인수의 유일한 정적 유형(선언된 유형)이 고려되고, 동적 유형(대부분의 특정 유형)은 고려되지 않습니다.

3. 2 단계 후에는 많아야 하나의 후보 메소드가 남습니다. 이것이 선택되는 메소드입니다.
4. 2단계 이후에 남아 있는 후보 메소드가 없는 경우, 오류가 발생합니다 (SQLSTATE 42884).

메소드 분석 예

다음은 성공적인 메소드 차수의 예입니다.

HEADOFSTATE의 부속 유형인 EMPEROR의 부속 유형으로 GOVERNOR 계층에 정의된 세 개의 구조화 유형에 대해 7개의 FOO 메소드가 있습니다. 이는 다음의 시그니처로 등록됩니다.

```
CREATE METHOD FOO (CHAR(5), INT, DOUBLE)  FOR HEADOFSTATE SPECIFIC FOO_1 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR HEADOFSTATE SPECIFIC FOO_2 ...
CREATE METHOD FOO (INT, INT, DOUBLE, INT) FOR HEADOFSTATE SPECIFIC FOO_3 ...
CREATE METHOD FOO (INT, DOUBLE, DOUBLE)   FOR EMPEROR     SPECIFIC FOO_4 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR EMPEROR     SPECIFIC FOO_5 ...
CREATE METHOD FOO (SMALLINT, INT, DOUBLE) FOR EMPEROR     SPECIFIC FOO_6 ...
CREATE METHOD FOO (INT, INT, DEC(7,2))    FOR GOVERNOR    SPECIFIC FOO_7 ...
```

메소드 참조는 다음과 같습니다(여기서, I1과 I2는 INTEGER 컬럼이고, D는 십진수 컬럼, E는 EMPEROR 컬럼입니다).

```
SELECT E..FOO(I1, I2, D) ...
```

알고리즘에 따르면...

FOO_7은 유형 GOVERNOR가 EMPEROR의 부속 유형(수퍼 유형이 아니라)이므로 후보에서 제거됩니다.

FOO_3은 매개변수의 개수가 틀리므로 후보에서 제거됩니다.

FOO_1과 FOO_6은 두 경우 모두 첫번째 인수(주제 인수가 아니라)가 첫번째 매개변수의 데이터 유형으로 승격될 수 없으므로 제거됩니다. 하나 이상의 후보가 남아 있으므로, 인수는 순서대로 고려되어야 합니다.

주제 인수에 대해, FOO_2는 수퍼 유형이고, FOO_4 및 FOO_5는 주제 인수와 일치합니다.

첫번째 인수에 대해, 나머지 메소드 FOO_4 및 FOO_5는 인수 유형과 정확히 일치합니다. 이 고려사항에서는 어떤 메소드도 제거되지 않았으므로 다음 인수가 검사되어야 합니다.

이 두 번째 인수의 경우, FOO_5는 정확히 일치하지만, FOO_4는 일치하지 않으므로, 고려 대상에서 제외됩니다. 이로서, FOO_5가 선택된 메소드로 남습니다.

메소드 호출

일단 메소드가 선택되어도, 메소드를 사용할 수 없는 이유가 여전히 있습니다.

각 메소드는 특정 데이터 유형을 가진 결과를 리턴하도록 정의됩니다. 이 결과 데이터 유형이 메소드가 호출된 문맥과 호환되지 않을 경우, 오류가 발생합니다. 예를 들어, STEP이라고 하는 다음 메소드들이 정의되고 각각 결과와 다른 데이터 유형을 가지고 있다고 가정합시다.

```
STEP(SMALLINT) FOR TYPEA RETURNS CHAR(5)
STEP(DOUBLE) FOR TYPEA RETURNS INTEGER
```

메소드 참조가 다음과 같은 경우(여기서 S는 SMALLINT 컬럼이고 TA는 TYPEA의 컬럼임),

```
SELECT 3 + TA..STEP(S) ...
```

인수 유형과 정확히 일치하는 것이 있으므로, 첫번째 STEP이 선택됩니다. 결과 유형이 추가 연산자의 인수에 필요한 숫자 유형이 아닌 CHAR(5)이므로 명령문에서 오류가 발생합니다.

선택된 메소드가 유형 보존 메소드일 경우 다음 사항에 유의하십시오.

- 함수 결정 다음의 정적 결과 유형은 메소드 호출의 주제 인수에 대한 정적 유형과 같습니다.

- 메소드가 호출될 때 동적 결과 유형은 메소드 호출의 주제 인수에 대한 동적 유형과 같습니다.

이는 유형 보존 메소드 정의에서 지정된 결과 유형의 부속 유형이 될 수 있습니다. 이것은 다시, 메소드가 처리될 때 실제로 리턴되는 동적 유형의 수퍼 유형이 될 수 있습니다.

메소드 호출의 인수가 선택된 메소드의 매개변수 데이터 유형과 정확히 일치하지 않는 경우, 인수는 칼럼에 대한 지정과 같은 규칙을 사용하여 실행시 매개변수의 데이터 유형으로 변환됩니다(109 페이지의 『지정 및 비교』 참조). 인수와 매개변수 사이에 정밀도, 스케일 또는 길이가 다른 경우가 여기에 속하며, 인수의 동적 유형이 매개변수의 정적 유형에 대한 부속 유형인 경우는 제외됩니다.

보수 바인딩 시멘틱

명령문이 처리될 때 함수, 메소드 및 데이터 유형이 분석되는 데이터베이스 내에 여러 상황이 있으므로 데이터베이스 관리 프로그램은 이 분석을 반복할 수 있어야 합니다. 이는 다음에서 적용됩니다.

- 패키지에 있는 정적 DML문,
- 뷰,
- 트리거,
- 점검 제한조건, 및
- SQL 루틴

패키지의 정적 DML문의 경우, 함수, 메소드 및 데이터 유형 참조는 바인드 조작 중에 분석됩니다. 뷰, 트리거 및 점검 제한조건에서의 함수, 메소드 및 데이터 유형 참조는 데이터베이스 오브젝트가 작성될 때 분석됩니다.

오브젝트에서의 함수 또는 메소드 참조에 대해 다시 함수 또는 메소드 분석이 수행될 경우, 새로운 함수나 메소드가 더 나은 일치를 보이지만 실제 실행 파일은 다른 조작을 수행하는 시그니처와 함께 추가된 경우 그 작동 방식을 변경할 수 있습니다. 마찬가지로, 오브젝트에서의 데이터 유형에 대해 다시 분석이 수행될 경우, 새로운 데이터 유형이 SQL 경로에도 있는 다른 스키마에서 같은 이름으로 추가된 경우 그 작동 방식을 변경할 수 있습니다. 이를 피하기 위해, 필요하다면 데이

터베이스 관리 프로그램이 *보수 바인딩 시멘틱*의 개념을 적용합니다. 이 개념은 함수 및 데이터 유형 참조가 바인드될 때와 같은 SQL 경로를 사용하여 분석되도록 합니다. 또한, 분석되는 동안 고려되는 함수, 메소드 및 데이터 유형의 작성 시간 소인은 [31]명령문이 바인드된 시간보다 늦지 않습니다.[32] 이 방식에서, 명령문이 원래 처리될 때 함수, 메소드 및 데이터 유형 분석 동안 고려된 함수 및 데이터 유형만 고려됩니다. 그러므로, 새로 작성되는 함수, 메소드 및 데이터 유형은 보존성이 있는 바인딩 시멘틱이 적용될 때 고려되지 않습니다.

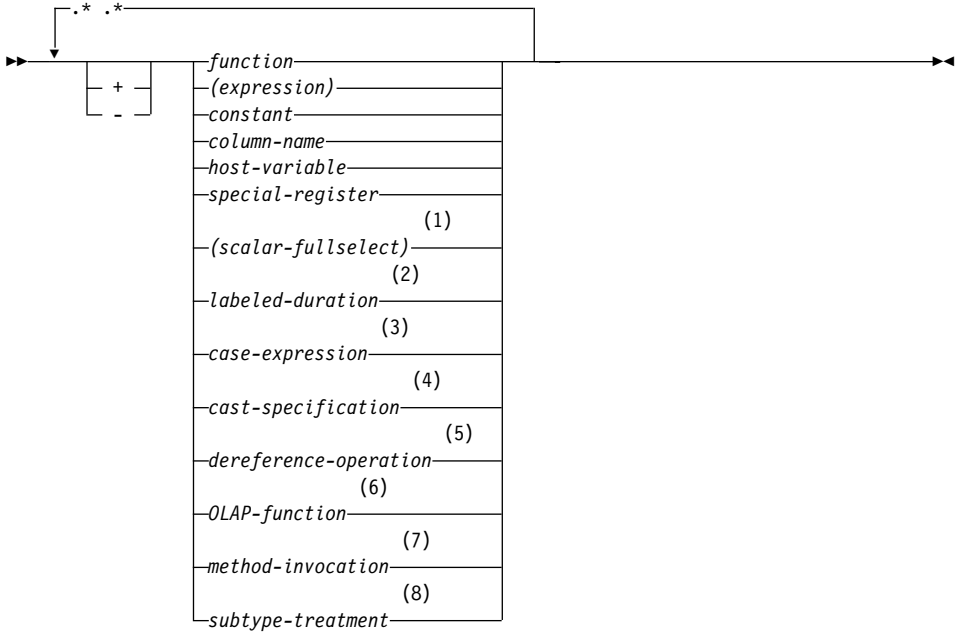
패키지에 있는 정적 DML의 경우에는, REBIND 명령(또는 이에 상응하는 API) 또는 BIND 명령(또는 이에 상응하는 API)을 발행하여 패키지를 내재적으로 또는 명시적으로 리바인드할 수 있습니다. 내재된 리바인드는 보존성이 있는 바인딩 의미론으로 함수, 메소드 및 데이터 유형을 분석하도록 항상 수행됩니다. REBIND 명령은 보존성이 있는 바인딩 의미론으로 분석하거나(RESOLVE CONSERVATIVE 옵션) 새로운 함수, 메소드 및 데이터 유형을 고려하여 분석(기본값으로 또는 RESOLVE ANY 옵션을 사용하여)하는 선택사항을 제공합니다.

표현식

표현식은 값을 지정합니다. 이것은 단 하나의 상수나 컬럼 이름으로 구성되는 간단한 값이 되거나, 더 복잡할 수 있습니다. 반복적으로 유사한 복잡한 표현식을 사용할 경우, SQL 함수를 사용하여 공통되는 표현식을 요약하는 것을 고려해 볼 수 있습니다. 726 페이지의 『CREATE FUNCTION (SQL 스칼라, 테이블 또는 행)』에서 자세한 설명을 참조하십시오.

31. 버전 6.1에서부터 추가된 내장 함수에는 데이터베이스 작성 또는 이주 시간을 기초로 하는 작성 시간소인이 있습니다.

32. 뷰의 경우, 보존성이 있는 바인딩은 또한 뷰의 컬럼이 뷰가 작성된 시간에 존재했던 컬럼과 같도록 합니다. 예를 들어, 선택 목록에서 별표를 사용해 정의된 뷰는 뷰가 작성된 후 기초가 되는 테이블에 추가된 어떠한 컬럼도 고려하지 않을 것입니다.



operator:

주:

1 191 페이지의 『스칼라 Fullselect』에서 자세한 설명을 참조하십시오.

2 191 페이지의 『레이블된 지속 기간』에서 자세한 설명을 참조하십시오.

3 199 페이지의 『CASE 표현식』에서 자세한 설명을 참조하십시오.

4 201 페이지의 『유형변환(CAST) 스펙』에서 자세한 설명을 참조하십시오.

5 205 페이지의 『참조해제(Dereference) 연산』에서 자세한 설명을 참조하십시오.

6 206 페이지의 『OLAP 함수』에서 자세한 설명을 참조하십시오.

7 212 페이지의 『메소드 호출』에서 자세한 설명을 참조하십시오.

8 214 페이지의 『부속 유형 처리』에서 자세한 설명을 참조하십시오.

9 ||는 CONCAT의 동의어로 사용됩니다.

연산자가 없는 표현식

연산자 없이 사용되는 경우, 표현식의 결과는 지정된 값이 됩니다.

예: `SALARY` `:SALARY` `'SALARY'` `MAX(SALARY)`

병합 연산자가 있는 표현식

병합 연산자(CONCAT)는 두 개의 피연산자를 연결하여 *문자열 표현식*을 구성합니다.

병합의 피연산자는 호환 가능한 문자열이어야 합니다. 2진 문자열은 FOR BIT DATA(SQLSTATE 42884)로 정의된 문자열을 포함하여 문자열과 병합될 수 없음을 기억하십시오. 호환에 대한 자세한 사항은 109 페이지의 표7의 호환 매트릭스를 참조하십시오.

피연산자가 널(NULL)인 경우, 결과는 널(NULL)이 될 수 있으며, 모두가 널(NULL)인 경우, 결과는 널(NULL)값입니다. 그렇지 않은 경우, 결과는 첫번째 피연산자와 다음의 두 번째 피연산자로 구성됩니다. 병합이 수행될 때 잘못 구성된 혼합 데이터에 대해 어떠한 점검도 수행하지 않음을 기억하십시오.

결과의 길이는 피연산자 길이의 합입니다.

결과의 데이터 유형과 길이 속성은 다음 표에 표시된 것처럼 피연산자의 내용으로 결정됩니다.

표 10. 병합된 피연산자의 데이터 유형과 길이

피연산자	병합된 길이 속성	결과
CHAR(A) CHAR(B)	<255	CHAR(A+B)
CHAR(A) CHAR(B)	>254	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
CHAR(A) LONG VARCHAR	-	LONG VARCHAR

표 10. 병합된 피연산자의 데이터 유형과 길이 (계속)

피연산자	병합된 길이 속성	결과
VARCHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
VARCHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
VARCHAR(A) LONG VARCHAR	-	LONG VARCHAR
LONG VARCHAR LONG VARCHAR	-	LONG VARCHAR
CLOB(A) CHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) VARCHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) LONG VARCHAR	-	CLOB(MIN(A+32K, 2G))
CLOB(A) CLOB(B)	-	CLOB(MIN(A+B, 2G))
GRAPHIC(A) GRAPHIC(B)	<128	GRAPHIC(A+B)
GRAPHIC(A) GRAPHIC(B)	>127	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
GRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
VARGRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
VARGRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
VARGRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
LONG VARGRAPHIC LONG VARGRAPHIC	-	LONG VARGRAPHIC
DBCLOB(A) GRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) VARGRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) LONG VARGRAPHIC	-	DBCLOB(MIN(A+16K, 1G))
DBCLOB(A) DBCLOB(B)	-	DBCLOB(MIN(A+B, 1G))
BLOB(A) BLOB(B)	-	BLOB(MIN(A+B, 2G))

이전 버전과의 호환의 경우, LONG 데이터 유형이 LOB 데이터 유형으로 자동 변환되는 일은 없습니다. 예를 들어, CHAR(200) 값과 완전한 LONG VARCHAR 값과의 병합은 CLOB 데이터 유형으로의 변환이 아닌 오류를 발생시킵니다.

결과의 코드 페이지는 추출(derived)된 코드 페이지이고, 129 페이지의 『문자열 변환 규칙』에서 설명한 바와 같이 피연산자의 코드 페이지에 의해 결정됩니다.

한 피연산자가 매개변수 표시문자가 될수 있습니다. 매개변수 표시문자가 사용되는 경우, 그 피연산자의 데이터 유형과 길이 속성은 비매개변수 표시문자 피연산자의 것과 같은 것으로 간주됩니다. 연산의 순서는 중첩된 병합의 경우 이러한 속성을 결정하기 위해 고려되어야 합니다.

예 1: FIRSTNME이 Pierre이고 LASTNAME이 Fermat일 경우, 다음은

```
FIRSTNME CONCAT ' ' CONCAT
LASTNAME
```

값 Pierre Fermat를 리턴합니다.

예 2: 주어지는 값:

- COLA VARCHAR(5), 값은 'AA'
- 길이가 5이고 값이 'BB '인 문자 호스트 변수로 정의된 :host_var
- COLC CHAR(5), 값 'CC'
- COLD CHAR(5), 값 'DDDDD'

COLA **CONCAT** :host_var **CONCAT** COLC **CONCAT** COLD의 값은 다음과 같습니다.

```
'AABB  CC  DDDDD'
```

데이터 유형은 VARCHAR, 길이 속성은 17이고, 결과 코드 페이지는 데이터베이스 코드 페이지입니다.

예 3: 주어지는 값:

```
COLA CHAR(10)
COLB VARCHAR(5)
```

표현식의 매개변수 표시문자는

```
COLA CONCAT COLB CONCAT ?
```

VARCHAR(15)로 간주되는데, 이는 COLA CONCAT COLB가 두 번째 CONCAT 연산의 첫번째 피연산자인 결과를 제공하는 것으로 먼저 평가되기 때문입니다.

사용자 정의 유형

소스 데이터 유형이 문자열 유형인 구별 유형일 경우에도, 사용자 정의 유형은 병합 연산자와 함께 사용할 수 없습니다. 병합하려면, 그 소스로 CONCAT 연산자를 갖는 함수를 작성하십시오. 예를 들어, 구별 유형 TITLE과 TITLE_DESCRIPTION이 있고 둘 모두에 VARCHAR(25) 데이터 유형이 있는 경우, 다음의 사용자 정의 함수 ATTACH는 이들을 병합하는 데 사용될 수 있습니다.

```
CREATE FUNCTION ATTACH (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

대체로, 병합 연산자는 사용자 정의 함수를 사용하여 오버로드되어 새로운 데이터 유형을 추가할 수 있습니다.

```
CREATE FUNCTION CONCAT (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

산술 연산자가 있는 표현식

산술 연산자가 사용되면, 표현식의 결과는 피연산자의 값에 연산자를 적용하여 추출한 값입니다.

모든 피연산자가 널(NULL)이 될 수 있거나 DFT_SQLMATHWARN을 yes로 설정하여 데이터베이스를 구성한 경우, 결과는 널(NULL)이 될 수 있습니다.

피연산자에 널(NULL)값이 있는 경우, 표현식의 결과는 널(NULL)값입니다.

산술 연산자는 부호가 있는 숫자 유형과 날짜 시간 유형에 적용될 수 있습니다(192 페이지의 『SQL에서의 날짜 시간 산술 연산』 참조). 예를 들어, USER+2는 유효하지 않습니다. 전래 함수는 부호가 있는 숫자 유형인 소스 유형이 있는 구별 유형에서 산술 연산에 대해 정의될 수 있습니다.

접두부 연산자 +(단일 플러스)는 이것의 피연산자를 변경하지 않습니다. 접두부 연산자 - (unary minus)는 0이 아닌 피연산자의 부호를 역으로 합니다. A의 데이

터 유형이 작은 정수일 경우, -A의 데이터 유형은 큰 정수입니다. 다음에 접두부 연산자가 오는 토큰의 첫번째 문자는 플러스나 마이너스의 부호가 될 수 없습니다.

infix 연산자 +, -, *, /는 덧셈, 뺄셈, 곱셈 및 나눗셈을 각각 지정합니다. 나눗셈에서 두 번째 피연산자의 값은 0이 될 수 없습니다. 이러한 연산자는 함수로 처리될 수도 있습니다. 따라서, 표현식 *"+"(a,b)*는 표현식 *a+b*.『연산자』함수와 동일합니다.

산술 오류

표현식을 처리하는 중 0으로 나누기 또는 숫자 오버플로우와 같은 산술 오류가 발생할 경우, 오류가 리턴되고 표현식을 처리하던 오류와 함께(SQLSTATE 22003 또는 22012) SQL문이 실패합니다.

산술 오류가 표현식에 대해 널(NULL) 값을 리턴시키고, 경고(SQLSTATE 01519 또는 01564)를 표시한 후 SQL문의 처리를 계속하도록, (DFT_SQLMATHWARN를 yes로 설정하여) 데이터베이스를 구성할 수 있습니다. 산술 오류가 널(NULL)로 취급될 경우, SQL문의 결과라고 할 수 있습니다. 다음의 이러한 경우에 대한 예입니다.

- 컬럼 함수의 인수인 표현식에서 발생한 산술 오류는 컬럼 함수의 결과를 정하는 데 있어 행이 무시되게 합니다. 산술 오류가 오버플로우일 경우, 이는 결과 값에 중대한 영향을 미칠 수 있습니다.
- WHERE절에 있는 술어의 표현식에서 발생한 산술 오류는 결과에 행이 포함되지 않게 할 수 있습니다.
- 점검 제한조건에 있는 술어의 표현식에서 발생한 산술 오류는 제한조건이 거짓이 아닐 수 있으므로 갱신이나 삽입 처리를 유발합니다.

이러한 유형의 영향을 받아들일 수 없는 경우, 산술 오류를 처리하기 위한 추가 조치를 취하여 수용 가능한 결과를 산출해야 합니다. 예를 들어 다음이 있습니다.

- 0으로 나누기가 있는지 점검하기 위한 case 표현식을 추가하고 그러한 경우에 대비하여 원하는 값을 설정합니다.
- 널(NULL)을 처리하기 위한 추가적인 술어를 추가합니다(널(NULL) 입력 불가능 컬럼의 점검 제한조건과 같이, 아래와 같이 되어

```
check(c1*c2 is not null and c1*c2>5000)
```

오버플로우에 대하여 제한조건이 위배되게 할 수 있습니다.).

두 개의 정수 피연산자

산술 연산자의 두 피연산자가 모두 정수이면, 2진으로 조작이 수행되며 결과는 *큰 정수*입니다. 두 피연산자(또는 둘 중의 하나)가 대 정수(big integer)가 아니면, 결과는 대 정수(big integer)입니다. 나눗셈의 나머지는 잃게 됩니다. 정수 산술 연산(단일 마이너스 포함)의 결과는 결과 유형 범위 내에 있어야 합니다.

정수와 십진수 피연산자

한 피연산자는 정수이고 다른 피연산자는 소수일 경우, 연산은 정밀도가 *p*이고 스케일이 0인 소수로 변환된 정수의 임시 사본을 사용하여 소수로 수행됩니다. *p*는 대 정수(big integer)의 경우 19이고 큰(large) 정수의 경우에는 11이며 작은(small) 정수의 경우에는 5입니다.

두 개의 소수 피연산자

피연산자 모두가 십진수인 경우, 연산은 십진수로 수행됩니다. 십진수 연산의 결과는 연산과 피연산자의 정밀도와 스케일에 따라 달라지는 십진수입니다. 연산이 덧셈이나 뺄셈이고 피연산자의 스케일이 같지 않은 경우, 연산은 피연산자 중 하나의 임시 사본으로 수행됩니다. 짧은 피연산자의 사본 뒤에 0이 첨부되어 확장되어 분수 부분이 긴 피연산자와 같은 디지트 수를 갖도록 합니다.

십진수 연산 결과의 정밀도는 31보다 클 수 없습니다. 십진수의 덧셈, 뺄셈, 곱셈의 결과는 정밀도가 31보다 큰 임시 결과에서 온 것입니다. 임시 결과의 정밀도가 31보다 크지 않은 경우, 마지막 결과는 임시 결과와 같습니다.

SQL에서의 소수 연산

다음 공식은 SQL에서의 십진수 연산 결과의 정밀도와 스케일을 정의합니다. 부호 *p*와 *s*는 첫번째 피연산자의 정밀도와 스케일을 나타내며, 부호 *p'*와 *s'*는 두 번째 피연산자의 정밀도와 스케일을 나타냅니다.

덧셈 및 뺄셈

정밀도는 min(31,max(*p-s,p'-s'*) + max(*s,s'*) + 1)입니다. 덧셈과 뺄셈의 결과 스케일은 max(*s,s'*)입니다.

곱셈

곱셉 결과의 정밀도는 min(31,*p+p'*)이고, 스케일은 min(31,*s+s'*)입니다.

나눗셈

나눗셈 결과의 정밀도는 31입니다. 스케일은 *31-p+s-s'*입니다. 스케일은 음수가 될 수 없습니다.

부동 소수점 피연산자

산술 연산자의 두 피연산자가 부동 소수점일 경우, 연산은 부동 소수점으로 수행되며 필요할 경우 피연산자가 먼저 배정밀도의 부동 소수로 변환됩니다. 따라서 표현식의 어느 한 요소라도 부동 소수일 경우, 표현식의 결과는 배정밀도의 부동 소수입니다.

부동 소수점 수와 정수가 들어 있는 연산은 배밀도 부동 소수점 수로 변환된 정수의 임시 사본으로 수행됩니다. 부동 소수점 수와 십진수가 들어 있는 연산은 배밀도 부동 소수점 수로 변환된 십진수의 임시 사본으로 수행됩니다. 부동 소수점 연산의 결과는 부동 소수점 수의 범위 내에 있어야 합니다.

피연산자의 사용자 정의 유형

사용자 정의 유형은 이의 소스 데이터 유형이 숫자라고 하여도 산술 연산자와 함께 사용할 수 없습니다. 산술 연산을 수행하려면, 소스로서 산술 연산자를 갖는 함수를 작성하십시오. 예를 들어, 구별 유형 INCOME 및 EXPENSES가 있고 이들이 모두 십진수(8,2) 데이터 유형인 경우, 다음에 오는 사용자 정의 함수 REVENUE는 다른 것에서 하나를 감산하는 데 사용됩니다.

```
CREATE FUNCTION REVENUE (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

대체로, -(마이너스) 연산자는 사용자 정의 함수를 사용하여 새로운 데이터 유형을 뺄셈하도록 오버로드될 수 있습니다.

```
CREATE FUNCTION "-" (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

스칼라 Fullselect

표현식에서 지원되는 *스칼라 fullselect*는 단일 컬럼 값으로 구성되는 단일 행을 리턴하는, 괄호로 묶인 fullselect입니다. fullselect가 행을 리턴하지 않을 경우, 표현식의 결과는 널(NULL) 값이 됩니다. 선택 목록 요소가 컬럼 이름 뿐이거나 참조 해제 연산뿐인 표현식인 경우, 결과 컬럼 이름은 해당 컬럼 이름에 기초합니다. 476 페이지의 『fullselect』에서 자세한 정보를 참조하십시오.

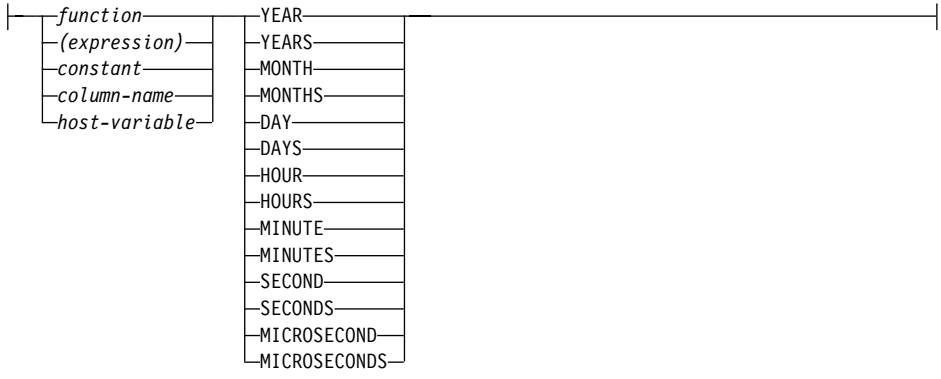
날짜 시간 연산 및 기간

날짜 시간값은 증가, 감소될 수 있습니다. 이러한 연산에는 *기간*이라는 십진수가 포함됩니다. 다음은 날짜 시간 연산에 대한 규칙의 스펙과 기간의 정의입니다.

기간은 시간의 간격을 나타내는 숫자입니다. 다음은 지속 기간의 네 가지 유형입니다.

레이블된 지속 기간

레이블된 기간::



*레이블된 지속 기간*은 7개의 지속기간 키워드인 YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS 또는 MICROSECONDS 중 하나가 뒤에 오는 숫자(표현식의 결과)로 표시되는 특정의 시간 단위를 나타냅니다.[33] 지정된 번호는 십진수(15,0) 숫자에 지정된 것처럼 변환됩니다. 레이블된 지속 기간은 산술 연산자의 피연산자로 사용될 수 있으며, 다른 피연산자는 데이터 유형 DATE,

33. 이 키워드들의 단수도 사용할 수 있습니다(YEAR, MONTH, DAY, HOUR, MINUTE, SECOND 및 MICROSECOND).

TIME 또는 TIMESTAMP의 값입니다. 그러므로, 표현식 HIREDATE + 2 MONTHS + 14 DAYS는 유효하며, 반면 표현식 HIREDATE +(2 MONTHS + 14 DAYS)는 유효하지 않습니다. 이러한 표현식의 경우, 레이블된 지속 기간은 2 MONTHS와 14 DAYS입니다.

날짜 기간

*날짜 기간(date duration)*은 십진수(8,0) 숫자로 표시되며, 연, 월, 일의 숫자를 나타냅니다. 적합하게 해석되려면, 숫자의 형식이 *yyyymmdd*여야 합니다. 여기서, *yyyy*는 년, *mm*은 월, *dd*는 일을 나타냅니다. [34] 표현식 HIREDATE - BRTHDATE에서처럼 다른 값에서 하나의 값을 뺀 결과가 날짜 기간입니다.

시간 기간

*시간 기간*은 십진수(6,0) 숫자로 표현되는 시, 분, 초를 나타냅니다. 적합하게 해석하려면, 숫자의 형식은 *hhmmss*여야 합니다. 여기서, *hh*는 시, *mm*은 분, *ss*는 초를 나타냅니다. 하나의 시간 값을 다른 시간 값에서 빼면 시간 기간이 됩니다.

시각 기간

*시각 기간*은 십진수(20,6) 숫자로 표현되는 년, 월, 일, 시, 분, 초 및 마이크로초를 나타냅니다. 적합하게 해석되려면, 숫자의 형식이 *yyyymmddhhmmss.zzzzzz*여야 합니다. 여기서 *yyyy*, *mm*, *dd*, *hh*, *mm*, *ss* 및 *zzzzzz*는 년, 월, 일, 시, 분, 초, 마이크로초를 각각 나타냅니다. 한 시각을 다른 시각에서 뺀 값은 시각 기간이 됩니다.

SQL에서의 날짜 시간 산술 연산

날짜 시간 값에서 수행할 수 있는 유일한 산술 연산이 덧셈과 뺄셈입니다. 날짜시간 값이 덧셈의 피연산자인 경우, 다른 피연산자는 기간이 됩니다. 날짜시간 값이 있는 덧셈 연산자의 사용에 대해 특별히 적용되는 규칙은 다음과 같습니다.

- 한 피연산자가 날짜인 경우, 다른 피연산자는 날짜 기간이거나 레이블된 지속 기간 YEARS, MONTHS 또는 DAYS이어야 합니다.
- 한 피연산자가 시간인 경우, 나머지 피연산자는 HOURS, MINUTES 또는 SECONDS의 시간 기간 또는 레이블된 지속 기간이어야 합니다.

34. 포맷의 기간은 십진수 데이터 유형을 나타냅니다.

- 한 피연산자가 시각인 경우, 나머지 피연산자는 기간이어야 합니다. 어느 유형의 기간도 유효합니다.
- 덧셈 연산자의 피연산자 중 어느것도 매개변수 표시문자가 될 수 없습니다.

날짜시간 값에 대한 뺄셈 연산자의 사용 규칙은 덧셈의 규칙과 같지 않습니다. 이는 날짜시간 값은 기간에서 뺄 수 없고, 두 날짜시간 값의 뺄셈은 날짜시간 값에서 기간을 빼는 것과 같지 않기 때문입니다. 날짜시간 값이 있는 뺄셈 연산자의 사용에 대해 특별히 적용되는 규칙은 다음과 같습니다.

- 첫번째 피연산자가 날짜인 경우, 두 번째 피연산자는 날짜, 날짜 기간, 날짜열 표현 또는 YEARS, MONTHS 또는 DAYS의 레이블된 지속 기간이어야 합니다.
- 두 번째 피연산자가 날짜인 경우, 첫번째 피연산자가 날짜 또는 날짜열 표현이어야 합니다.
- 첫번째 피연산자가 시간인 경우, 두 번째 피연산자는 시간, 시간 기간, 시간열 표현 또는 HOURS, MINUTES 또는 SECONDS의 레이블된 지속 기간이어야 합니다.
- 두 번째 피연산자가 시간인 경우, 첫번째 피연산자가 시간 또는 시간의 문자열 표현이어야 합니다.
- 첫번째 피연산자가 시각인 경우, 두 번째 피연산자는 시각, 시각열 표현 또는 기간이어야 합니다.
- 두 번째 피연산자가 시각인 경우, 첫번째 피연산자가 시각 또는 시각열 표현이어야 합니다.
- 뺄셈 연산자의 어느 피연산자도 매개변수 표시문자가 될 수 없습니다.

날짜 산술 연산

날짜는 빼거나, 증가 또는 감소될 수 있습니다.

날짜 뺄셈: 한 날짜(DATE1)에서 다른 날짜(DATE2)를 뺄셈한 결과는 두 날짜 간의 연, 월, 일수를 지정하는 날짜 기간입니다. 결과의 데이터 유형은 D십진수(8,0)입니다. DATE1이 DATE2보다 같거나 큰 경우, DATE2가 DATE1에서 빼집니

다. 그러나 DATE1이 DATE2보다 작은 경우, DATE1이 DATE2에서 빼지고, 결과의 부호는 음수가 됩니다. 다음 프로시듀어는 연산 결과 = DATE1 - DATE2에 포함되는 단계를 보여줍니다.

```
If DAY(DATE2) <= DAY(DATE1)
     then DAY(RESULT) = DAY(DATE1) - DAY(DATE2).

If DAY(DATE2) > DAY(DATE1)
     then DAY(RESULT) = N + DAY(DATE1) - DAY(DATE2)
        where N = the last day of MONTH(DATE2).
           MONTH(DATE2) is then incremented by 1.

If MONTH(DATE2) <= MONTH(DATE1)
     then MONTH(RESULT) = MONTH(DATE1) - MONTH(DATE2).

If MONTH(DATE2) > MONTH(DATE1)
     then MONTH(RESULT) = 12 + MONTH(DATE1) - MONTH(DATE2).
        YEAR(DATE2) is then incremented by 1.

YEAR(RESULT) = YEAR(DATE1) - YEAR(DATE2).
```

예를 들어, DATE('3/15/2000') - '12/31/1999'의 결과는 00000215입니다. (또는 0년, 2개월, 15일의 기간입니다.)

날짜의 증가와 감소: 기간을 날짜에 더하거나, 날짜에서 기간을 뺀 결과는 그 자체가 날짜입니다(이러한 연산에서, 월은 달력의 한 페이지와 같습니다. 날짜에 월을 더하면 이것은 날짜가 나타나는 페이지부터 시작하는 달력의 페이지를 넘기는 것과 같습니다.). 결과는 날짜 1월 1일, 0001과 12월 31일, 9999 사이에 있어야 합니다.

년의 기간이 더해지거나 빼지면, 날짜의 년 위치에만 영향을 줍니다. 결과가 윤년이 아닌 해의 2월 29일 경우를 제외하면, 월은 변경되지 않습니다. 이런 경우, 일은 28로 변경되고, SQLCA에 있는 경고 표시기가 조정을 지시하도록 표시됩니다.

유사하게, 월의 기간이 더해지거나 빼지는 경우, 필요하면 월과 년에만 영향을 줍니다. 날짜의 일 위치는 결과가 유효하지 않는 한(예를 들면, 9월 31처럼) 변경되지 않습니다. 이런 경우, 일은 월의 마지막 일로 설정되고, SQLCA에 있는 경고 표시기가 조정을 지시하도록 설정됩니다.

물론 기간의 덧셈이나 뺄셈은 날짜의 일 위치에 영향을 미치고, 잠재적으로 월과 년에도 영향을 줍니다.

날짜 기간(음, 양에 관계없음)은 날짜에 더해지거나 빼질 수 있습니다. 레이블된 지속 기간에서 처럼, 결과는 유효한 날짜이고, 경고 표시기는 월 종료의 조정이 필요할 때마다 설정됩니다.

양수의 날짜 기간이 날짜에 더해지거나, 음수의 날짜 기간이 날짜에서 빼지면, 날짜는 연, 월, 일의 지정된 숫자만큼 증가됩니다. 그러므로, DATE1 + X(여기서, X는 음수 십진수(8,0))는 다음 표현식과 같은 값입니다.

```
DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS.
```

양수의 날짜 기간이 날짜에서 빼지거나, 음수의 날짜 기간이 날짜에 더해지면, 날짜는 연, 월, 일의 지정된 숫자만큼 감소됩니다. 그러므로, DATE1 - X(여기서, X는 음수 십진수(8,0))는 다음 표현식과 같은 값입니다.

```
DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS.
```

날짜에 기간 추가할 때, 주어진 날짜에 한 달을 더하면 그 날짜가 다음 달에 없지 않은 한, 1개월 이후의 같은 날짜로 됩니다. 이런 경우, 날짜는 마지막 월의 마지막 날짜로 됩니다. 예를 들어, 1월 28일에 1월을 더하면 2월 28일이 되고, 1월 29, 30, 31일에 1개월을 더하면 모두 2월 28일이 되고, 윤년의 경우는 2월 29일이 됩니다.

주: 날짜에 하나 이상의 월을 더하고, 월과 같은 수를 결과에서 빼면, 마지막 날짜는 원래의 날짜와 같지 않습니다.

시간 산술 연산

시간은 빼거나, 증가 또는 감소될 수 있습니다.

시간 빨셈: 한 시간(TIME1)에서 다른 시간(TIME2)를 뺀 결과는 두 시간간의 시, 분, 초를 나타내는 시간 기간입니다. 결과의 데이터 유형은 십진수(6,0)입니다.

TIME1이 TIME2보다 같거나 큰 경우, TIME2가 TIME1에서 빼집니다.

그러나 TIME1이 TIME2보다 작은 경우, TIME1이 TIME2에서 빼지고, 결과의 부호는 음수가 됩니다. 다음 프로시듀어는 연산 결과 = TIME1 - TIME2에 포함되는 단계를 보여줍니다.

```
If SECOND(TIME2) <= SECOND(TIME1)
     then SECOND(RESULT) = SECOND(TIME1) - SECOND(TIME2).
```

```
If SECOND(TIME2) > SECOND(TIME1)
     then SECOND(RESULT) = 60 + SECOND(TIME1) - SECOND(TIME2).
        MINUTE(TIME2) is then incremented by 1.

If MINUTE(TIME2) <= MINUTE(TIME1)
     then MINUTE(RESULT) = MINUTE(TIME1) - MINUTE(TIME2).

If MINUTE(TIME1) > MINUTE(TIME1)
     then MINUTE(RESULT) = 60 + MINUTE(TIME1) - MINUTE(TIME2).
        HOUR(TIME2) is then incremented by 1.

HOUR(RESULT) = HOUR(TIME1) - HOUR(TIME2).
```

예를 들어, TIME('11:02:26') - '00:32:56'의 결과는 102930입니다. (10시간, 29분, 30초의 기간입니다.)

시간의 증가와 감소: 기간을 시간에 더하거나 시간 기간에서의 뺄셈 결과는 그 자체가 시간입니다. 시간의 오버플로우와 언더플로우는 무시되므로, 결과는 언제나 시간입니다. 시간의 기간이 더해지거나 빼지는 경우, 시간의 시 위치에만 영향을 줍니다. 분과 초는 변경되지 않습니다.

유사하게, 분의 기간이 더해지거나 빼지는 경우, 필요하면 분과 시에만 영향을 줍니다. 시간의 초 위치는 변경되지 않습니다.

물론 초의 덧셈이나 뺄셈은 시간의 초 위치에 영향을 미치고, 잠재적으로 분과 시에도 영향을 줍니다.

음양에 관계없이 시간 기간은 시간에 더해지고, 시간에서 빼질 수 있습니다. 결과는 지정된 시, 분, 초 수만큼씩, 이 순서대로 증가되거나 감소된 시간입니다. TIME1 + X(여기서, "X"는 십진수(6,0))는 다음 표현식과 같은 값입니다.

```
TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
```

주: 시간 '24:00:00'이 유효한 시간으로 승인되어도, 이 시간은 지속기간 피연산자가 0(예: time('24:00:00')±0 seconds = '00:00:00')일 경우에도 시간 덧셈 및 뺄셈의 결과로 리턴되지 않습니다.

시각 연산

시각은 빼거나, 증가 또는 감소될 수 있습니다.

시각 뺄셈: 한 시각(TS1)에서 다른 시각(TS2)을 뺀 결과는 두 시각간의 시, 분, 초를 나타내는 시각 기간입니다. 결과의 데이터 유형은 십진수(20,6)입니다.

TS1이 TS2보다 같거나 큰 경우, TS2가 TS1에서 빼집니다. 그러나 TS1이 TS2보다 작은 경우, TS1이 TS2에서 빼지고, 결과의 부호는 음수가 됩니다. 다음 프로시듀어는 연산 결과 = TS1 - TS2에 포함되는 단계를 보여줍니다.

```
If MICROSECOND(TS2) <= MICROSECOND(TS1)
     then MICROSECOND(RESULT) = MICROSECOND(TS1) -
        MICROSECOND(TS2).

If MICROSECOND(TS2) > MICROSECOND(TS1)
     then MICROSECOND(RESULT) = 1000000 +
        MICROSECOND(TS1) - MICROSECOND(TS2)
           and SECOND(TS2) is incremented by 1.
```

시각의 초와 분은 시간 뺄셈에 대한 규칙에서 지정된 것처럼 빼집니다.

```
If HOUR(TS2) <= HOUR(TS1)
     then HOUR(RESULT) = HOUR(TS1) - HOUR(TS2).

If HOUR(TS2) > HOUR(TS1)
     then HOUR(RESULT) = 24 + HOUR(TS1) - HOUR(TS2)
        and DAY(TS2) is incremented by 1.
```

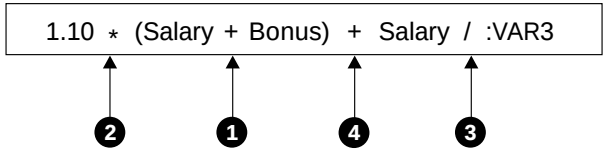
시각의 날짜 부분은 날짜 뺄셈 규칙에서 지정된 것처럼 빼집니다.

시각의 증가와 감소: 기간을 시각에 더하거나, 시각 기간에서의 뺀 결과는 그 자체가 시각입니다. 날짜와 시간 산술 연산은 이전에 정의된 것처럼 수행됩니다. 이때, 시간의 오버플로우나 언더플로우는 결과의 일부분에 영향을 주지만 유효한 날짜 범위 내에 있어야 합니다. 마이크로초는 초로 오버플로우됩니다.

연산 순서

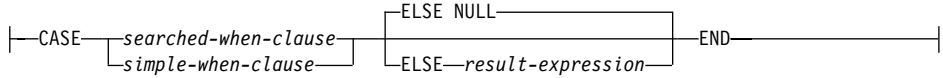
괄호안의 표현식과 참조해제 조작이 먼저 왼쪽에서 오른쪽으로 평가됩니다.[35] 평가 순서가 괄호로 지정되지 않은 경우, 곱셈과 나눗셈 이전에 접두부 연산자가 적용되고 덧셈과 뺄셈 이전에 곱셈과 나눗셈이 적용됩니다. 같은 우선순위 레벨에 있는 연산자의 경우는 왼쪽에서 오른쪽으로 적용됩니다.

35. 괄호는 부속 선택 명령문, 검색 조건 및 함수에 사용되기도 한다는 점에 유의하십시오. 그러나, SQL문 내에서 그룹 선택에 임시로 사용해서는 안됩니다.

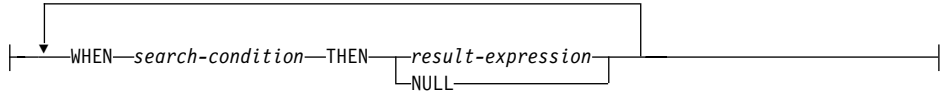

1.10 * (Salary + Bonus) + Salary / :VAR3
2
1
4
3

CASE 표현식

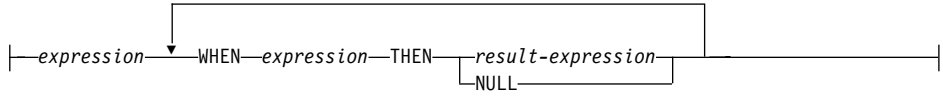
case-expression:



searched-when-clause:



simple-when-clause:



CASE 표현식에서는 하나 이상의 조건을 평가하여 그것을 기준으로 표현식을 선택할 수 있습니다. 일반적으로, CASE 표현식의 값은 참으로 평가된 첫번째 경우가 따르는 *결과 표현식*의 값입니다. 참인 경우가 없고, ELSE 키워드가 있는 경우, 결과는 *결과 표현식*의 값이나 널(NULL)이 됩니다. 참인 경우가 없고, ELSE 키워드도 없는 경우, 결과는 널(NULL)이 됩니다. CASE가 알 수 없음(널(NULL)로 인해)으로 평가되는 경우, 그 경우는 참이 아니므로 거짓으로 평가된 경우와 같이 처리됩니다.

CASE 표현식이 VALUES절, IN 술어, GROUP BY절 또는 ORDER BY절에 있는 경우, searched-when절의 *search-condition*은 규정화된 술어, fullselect를 사용하는 IN 술어 또는 EXISTS 술어가 될 수 없습니다(SQLSTATE 42625).

*simple-when절*을 사용할 때, 첫번째 *WHEN* 앞에 있는 *표현식* 값은 뒤에 WHEN 키워드가 오는 *표현식* 값과 같은지 테스트됩니다. 첫번째 *WHEN* 키워드 앞에 오는 *표현식*의 데이터 유형은 뒤에 WHEN 키워드가 오는 각각의 *표현식* 데이터 유

형과 비교될 수 있어야 합니다. *simple-when*절에서 첫번째 *WHEN* 키워드 앞에 오는 *표현식*은 변형 함수 또는 외부 조치가 있는 함수를 포함할 수 없습니다 (SQLSTATE 42845).

*결과 표현식*은 뒤에 THEN이나 ELSE 키워드가 오는 *표현식*입니다. CASE 표현식에는 적어도 하나의 *결과 표현식*이 있어야 합니다(모든 경우에 NULL은 지정될 수 없습니다)(SQLSTATE 42625). 모든 *결과 표현식*에는 호환 가능한 데이터 유형(SQLSTATE 42804)이 있어야 하며, 여기서 결과의 속성은 125 페이지의 『결과 데이터 유형 규칙』에 따라 결정됩니다.

예:

- 부서 번호의 첫번째 문자가 조직에서의 디비전일 경우, CASE 표현식을 사용하여 각 사원이 속하는 디비전의 완전한 이름을 나열할 수 있습니다.

```
SELECT EMPNO, LASTNAME,
       CASE SUBSTR(WORKDEPT,1,1)
       WHEN 'A' THEN 'Administration'
       WHEN 'B' THEN 'Human Resources'
       WHEN 'C' THEN 'Accounting'
       WHEN 'D' THEN 'Design'
       WHEN 'E' THEN 'Operations'
         END
FROM EMPLOYEE;
```

- 교육 년수는 EMPLOYEE 테이블에서 교육 레벨을 결정하는 데 사용됩니다. CASE 표현식은 이들을 그룹으로 분리하고, 교육 레벨을 표시하는 데 사용될 수 있습니다.

```
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
 CASE
       WHEN EDLEVEL < 15 THEN 'SECONDARY'
       WHEN EDLEVEL < 19 THEN 'COLLEGE'
       ELSE 'POST GRADUATE'
         END
                              FROM EMPLOYEE
```

- CASE문의 다른 재미있는 예는 0으로 나누는 오류를 방지하는 것입니다. 예를 들어, 다음 코드는 수수료로 수입의 25% 이상을 벌지만 수수료를 완전히 지불받지는 않는 사원을 탐색합니다.

```
SELECT EMPNO, WORKDEPT, SALARY+COMM FROM EMPLOYEE
WHERE(CASE WHEN SALARY=0 THEN NULL
           ELSE COMM/SALARY
           END) > 0.25;
```

- 다음의 CASE 표현식은 같습니다.

```
SELECT LASTNAME,
 CASE
 WHEN LASTNAME = 'Haas' THEN 'President'
 ...

SELECT LASTNAME,
 CASE LASTNAME
 WHEN 'Haas' THEN 'President'
 ...
```

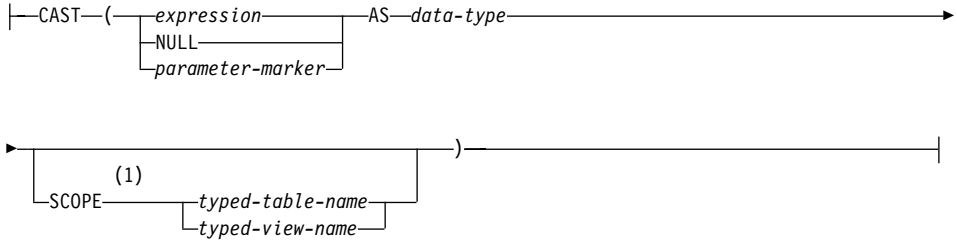
CASE에서 제공하는 기능의 부속 집합을 처리하도록 지정된 두 개의 스칼라 함수 NULLIF와 COALESCE가 있습니다. 표11은 CASE나 이러한 함수를 사용한 동일한 표현식을 보여줍니다.

표 11. 동일한 CASE 표현식

표현식	동일한 표현식
CASE WHEN e1=e2 THEN NULL ELSE e1 END	NULLIF(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE e2 END	COALESCE(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE COALESCE(e2,...,eN) END	COALESCE(e1,e2,...,eN)

유형변환(CAST) 스펙

cast-specification:



주:

1 SCOPE절은 REF 데이터 유형에만 적용됩니다.

CAST 스펙은 유형변환(CAST) 피연산자(첫번째 피연산자) CAST를 *데이터 유형*에 지정된 유형으로 리턴합니다.

표현식

유형변환(cast) 피연산자가 표현식일 경우(매개변수 표시문자나 널(NULL)이 아님), 결과는 지정된 목표 *데이터 유형*으로 변환된 인수 값입니다.

지원되는 변환은 108 페이지의 표6에 표시되며, 여기서 첫번째 컬럼은 cast 피연산자의 데이터 유형(원시 데이터 유형)을 나타내며, 맨위에 있는 데이터 유형은 CAST 스펙의 목표 데이터 유형을 나타냅니다. 변환이 지원되지 않는 경우, 오류가 발생합니다(SQLSTATE 42846).

문자열(CLOB는 제외)을 길이가 다른 문자열로 변환하면 뒤에 오는 공백 이외의 값을 절단한 경우, 경고(SQLSTATE 01004)가 표시됩니다. 그래픽 문자열(DBCLOB 제외)을 다른 길이의 그래픽 문자열로 변환할 때, 뒷 공백 이외의 절단이 발생할 경우, 경고(SQLSTATE 01004)가 표시됩니다. BLOB, CLOB 및 DBCLOB 변환(cast) 피연산자의 경우, 문자들이 절단되면 경고가 표시됩니다.

NULL

유형변환(cast) 피연산자가 키워드 널(NULL)일 경우, 결과는 지정된 *데이터 유형*을 갖는 널(NULL) 값입니다.

매개변수 표시문자

매개변수 표시문자(물음표로 표시됨)는 일반적으로 표현식으로 간주되지만, 이것에 특별한 의미가 있는 경우 분리되어 설명됩니다. 유형변환(cast) 피연산자가 *매개변수 표시문자*일 경우, 지정된 *데이터 유형*은(문자열에 대한 저장영역 지정을 통해) 그러한 데이터 유형으로 교체할 수 있음을 나타내는 보증으로 간주됩니다. 이러한 매개변수 표시문자는 *입력된 매개변수 표시문자*로 간주됩니다. 입력된 매개변수 표시문자는 함수 차수의 목적으로 입력된 다른 값, 선택 목록의 DESCRIBE 또는 컬럼 지정에 대한 것처럼 처리됩니다.

데이터 유형

기존의 데이터 유형 이름. 유형 이름이 규정화되지 않은 경우, SQL 경로는 데이터 유형 해석을 수행하는 데 사용됩니다. 길이나 정밀도 및 스케일과 같은 관련 속성을 갖는 데이터 유형에는 *데이터 유형* 지정시 이러한 속성이 포함되어야 합니다(지정되지 않을 경우, CHAR의 기본값은 길이 1로 설정되며

DECIMAL의 기본값은 정밀도 5, 스케일 0으로 설정됩니다.). 지원되는 데이터 유형에 대한 제한사항은 지정된 유형변환(cast) 피연산자를 기준으로 합니다.

- *표현식*인 유형변환(cast) 피연산자의 경우, 유형변환(cast) 피연산자의 데이터 유형(소스 데이터 유형)에 따라 지원되는 목표 데이터 유형에 대해서는 106 페이지의 『데이터 유형간의 변환』에서 참조하십시오.
- 키워드가 널(NULL)인 유형변환(cast) 피연산자의 경우, 기존의 데이터 유형이 사용될 수 있습니다.
- 매개변수 표시문자인 유형변환(cast) 피연산자의 경우, 목표 데이터 유형은 기존의 데이터 유형이 될 수 있습니다. 데이터 유형이 사용자 정의 구별 유형일 경우, 매개변수 표시문자를 사용하는 응용프로그램은 사용자 정의 구별 유형의 소스 데이터 유형을 사용합니다. 데이터 유형이 사용자가 정의하는 구조화 유형일 경우, 매개변수 표시문자를 사용하는 응용프로그램은 사용자가 정의하는 구조화 유형에 대한 TO SQL 변환 함수의 입력 매개변수 유형을 사용합니다.

SCOPE

데이터 유형이 참조 유형인 경우, 참조의 목표 뷰 또는 목표 테이블을 식별하는 영역이 지정될 수 있습니다.

입력된 테이블 이름

입력된 테이블의 이름. 테이블은 이미 존재해야 합니다(SQLSTATE 42704). 유형변환(cast)은 *데이터 유형* REF(*S*)로 되어야 합니다. 여기서 *S*는 *입력된 테이블 이름* 유형입니다(SQLSTATE 428DM).

입력된 뷰 이름

입력된 뷰의 이름. 뷰는 반드시 존재하거나, 작성중인 뷰와 동일한 이름을 가져야 합니다. 이 뷰에는 뷰 정의 일부로서 유형변환(cast)이 포함되어 있습니다(SQLSTATE 42704). 유형변환(cast)은 *데이터 유형* REF(*S*)로 되어야 합니다. 여기서 *S*는 *입력된 뷰 이름* 유형입니다(SQLSTATE 428DM).

숫자 데이터가 문자 데이터로 변환되면, 결과 데이터 유형은 고정 길이 문자열(291 페이지의 『CHAR』 참조)입니다. 문자 데이터가 숫자로 변환되면, 결과 데이터 유형은 지정된 숫자의 유형에 따라 달라집니다. 예를 들어, 정수로 변환하는 경우, 큰 정수가 됩니다(344 페이지의 『INTEGER』 참조).

예:

- 응용프로그램이 EMPLOYEE 테이블의 SALARY(십진수(9,2)로 정의됨)의 정수 부분에만 적용됩니다. 사원 번호와 정수값 SALARY를 포함한 다음 조회가 준비됩니다.

```
SELECT EMPNO, CAST(SALARY AS INTEGER) FROM EMPLOYEE
```

- SMALLINT로 정의되고 PERSONNEL 테이블에서 컬럼 AFE를 작성하는 데 사용되는 T_AGE 구별 유형이 존재한다고 가정하고 또 한 정수로 정의되고 PERSONNEL 테이블에서 컬럼 RETIRE_YEAR를 작성하는 데 사용되는 R_YEAR 구별 유형이 존재한다고 가정합시다. 다음 갱신 명령문이 준비됩니다.

```
UPDATE PERSONNEL SET RETIRE_YEAR =?
                 WHERE AGE = CAST( ? AS T_AGE)
```

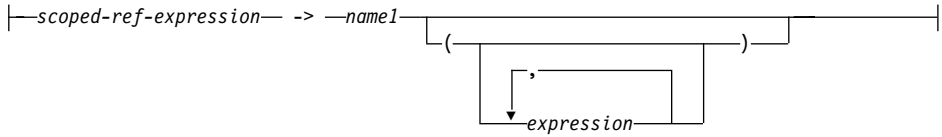
  첫번째 매개변수는 응용프로그램에서 이 매개변수 표시문자에 대해 정수를 사용할 지라도 R_YEAR의 데이터 유형을 갖는 미입력 매개변수 표시문자입니다. 이것은 지정(assignment)이므로 명시적인 CAST 스펙을 필요로 하지 않습니다.

  두 번째 매개변수 표시문자는 구별 유형 T_AGE로 변환되고 입력된 매개변수 표시문자입니다. 이는 호환되는 데이터 유형에 대해 비교가 수행되어야 하는 요구사항을 만족시킵니다. 응용프로그램에서는 소스 데이터 유형(SMALLINT)을 이용하여 이 매개변수 표시문자를 처리할 것입니다.

  이 명령문이 성공적으로 처리되려면 함수 경로에 두 개의 구별 유형이 정의된 스키마(들)의 스키마 이름이 포함되어 있어야 합니다.

참조해제(Dereference) 연산

참조해제(dereference) 연산:



범위 지정된 참조 표현식의 범위는 목표 테이블 또는 뷰라고 하는 테이블이나 뷰입니다. 범위 지정된 참조 표현식은 목표 *행*을 식별합니다. 목표 *행*은 오브젝트 식별자(OID) 컬럼 값이 참조 표현식과 일치하는 목표 테이블이나 뷰(또는, 해당되는 하위 테이블 또는 하위 뷰의 목표 테이블이나 뷰)의 행입니다. OID 컬럼에 대해서는 800 페이지의 『CREATE TABLE』에서 자세한 내용을 참조하십시오. 참조 취소 조작은 목표 행의 컬럼에 액세스하거나, 메소드의 주제로 목표 행을 사용하여 메소드를 호출하기 위해 사용할 수 있습니다. 참조 취소 조작의 결과는 항상 널(NULL)입니다. 참조해제 연산은 다른 모든 연산자에 대해 우선권을 가집니다.

영역지정 참조 표현식

영역을 가진 참조 유형인 표현식(SQLSTATE 428DT). 표현식이 호스트 변수인 경우, 참조에 영역을 제공하기 위해서는 매개변수 표시문자나 다른 비영역 참조 유형 값, 즉 SCOPE절을 가진 CAST 스펙이 필요합니다.

이름1

규정되지 않은 식별자를 지정합니다.

name1 다음에 어떤 괄호도 없고, *name1*이 목표 유형의 속성 이름과 일치할 경우, 참조 취소 조작의 값은 목표 행에서 명명된 컬럼의 값입니다. 이 경우, 널(NULL) 입력 가능하게 된 컬럼의 데이터 유형은 참조 취소 조작의 결과 유형을 판별합니다. 오브젝트 식별자가 참조 표현식과 일치하는 목표 행이 전혀 없을 경우, 참조 취소 조작의 결과는 널(NULL)입니다. 참조 취소 조작이 선택 목록에서 사용되고 표현식의 부분으로 포함되지 않은 경우, *name1*이 결과 컬럼 이름이 됩니다.

name1 다음에 괄호가 있고, *name1*이 목표 유형의 속성 이름과 일치하지 않을 경우, 참조 취소 조작은 메소드 호출로 처리됩니다. 호출된 메소드의 이름이 *name1*입니다. 메소드의 주제는 목표 행으로, 구조화 유형의 인스턴스로 간

주됩니다. 오브젝트 식별자가 참조 표현식과 일치하는 목표 행이 전혀 없을 경우, 메소드의 주제는 목표 유형의 널(NULL) 값입니다. 괄호 내의 표현식은(있을 경우) 메소드 호출의 나머지 매개변수를 제공합니다. 보통 프로세스는 메소드 호출의 분석에 사용됩니다. 널(NULL) 입력 가능하게 된 선택된 메소드의 결과 유형은 참조 취소 조작의 결과 유형을 판별합니다.

참조해제 연산을 사용하는 명령문의 권한 부여 ID에는 *영역지정 참조 표현식*의 목표 테이블에 대한 SELECT 특권이 있어야 합니다(SQLSTATE 42501).

참조 취소 조작은 데이터베이스의 값을 수정할 수 없습니다. 참조 취소 조작이 mutator 메소드를 호출하기 위해 사용될 경우, mutator 메소드는 목표 행의 사본을 수정하고 그 사본을 리턴하여, 데이터베이스는 변경되지 않도록 합니다.

예:

- 속성 DEPTNAME을 포함하는 유형에 기초한 입력된 테이블로 영역 지정된 참조 유형인 DEPTREF라는 컬럼이 들어 있는 EMPLOYEE 테이블이 있다고 가정합니다. EMPLOYEE 테이블의 DEPTREF 값들은 DEPTREF 컬럼의 목표 테이블에 있는 OID 컬럼 값과 일치해야 합니다.

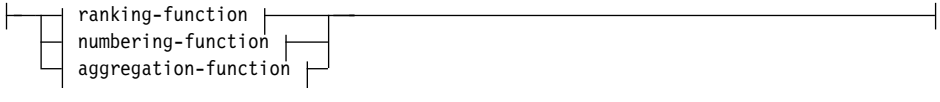
```
SELECT EMPNO, DEPTREF->DEPTNAME
                        FROM EMPLOYEE
```

- 이전 예에서와 같은 테이블을 사용할 경우, 주제 매개변수로 목표 행을, 추가 매개변수로 '1997'을 사용하여, 참조 취소 조작으로 BUDGET 메소드를 호출하십시오.

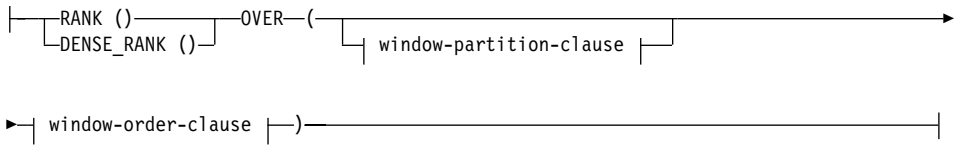
```
SELECT EMPNO, DEPTREF->BUDGET('1997')
  AS DEPTBUDGET97
  FROM EMPLOYEE
```

OLAP 함수

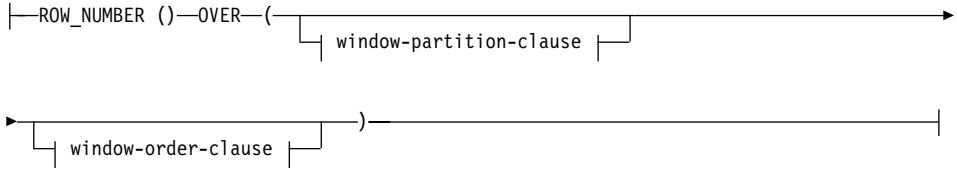
OLAP-function:



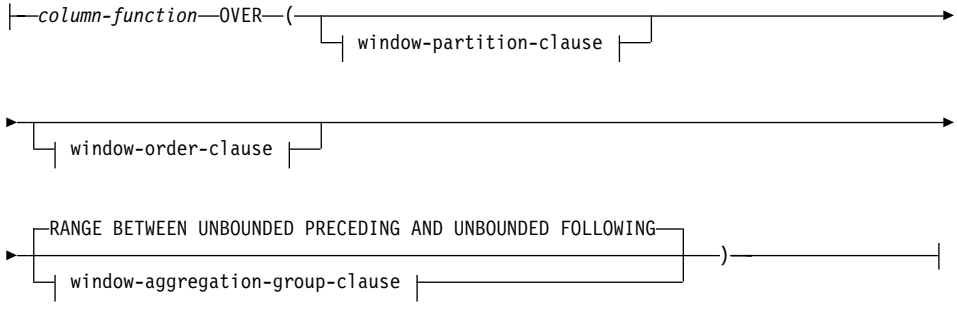
ranking-function:



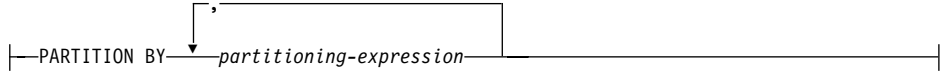
numbering-function:



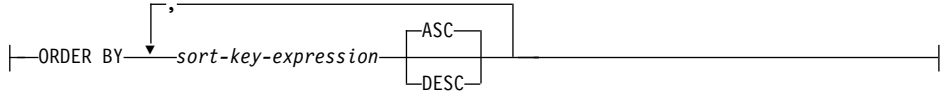
aggregation-function:



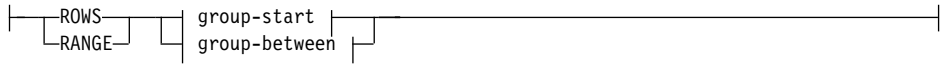
window-partition-clause:



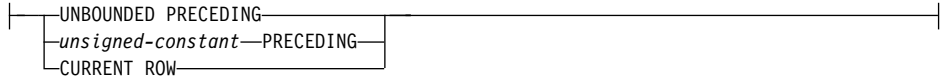
window-order-clause:



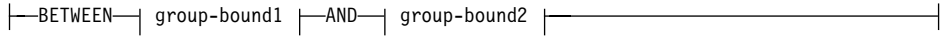
window-aggregation-group-clause:



group-start:



group-between:



group-bound1:

group-bound2:

OLAP(On-Line Analytical Processing) 함수는 조회 결과로 순위지정, 행 번호매김 및 기존의 컬럼 함수 정보를 스칼라 값으로 리턴할 수 있는 기능을 제공합니다. OLAP 함수는 선택 목록이나 선택 명령문의 ORDER BY 절에서 표현식에 포함될 수 있습니다(SQLSTATE 42903). OLAP 함수는 컬럼 함수의 인수로 사용할 수 없습니다(SQLSTATE 42607). OLAP 함수가 적용되는 조회 결과는 OLAP 함수를 포함하는 가장 안쪽의 부속 선택에 대한 결과 테이블입니다.

OLAP 함수를 지정할 때, 함수가 적용되는 행과 그 순서를 정의하는 창이 지정됩니다. 컬럼 함수와 함께 사용할 경우, 적용 가능한 행은 현재 행 앞과 뒤에 있는 행들의 범위나 행 수로, 현재 행에 상대적으로 더 세분화될 수 있습니다. 예를 들어, 월별 파티션 내에서, 평균은 이전 3개월 동안 계산될 수 있습니다.

순위 지정 함수는 창 내에서 행의 일반 순위를 계산합니다. 창 내에서의 순서 지정에 대해 구별되지 않는 행들에는 같은 순위가 지정됩니다. 순위 지정의 결과는 중복 값을 초래하는 숫자들에 간격을 두거나 두지 않고 정의할 수 있습니다.

RANK를 지정할 경우, 행의 순위는 행 앞에 있는 행 수에 1을 더해서 정의됩니다. 그러므로, 순서 지정에 대해 두 개 이상의 행이 구별되지 않으면, 순차 순위 지정에서 하나 이상의 간격이 생기게 됩니다.

DENSE_RANK[36]를 지정할 경우, 행의 순위는 순서에 대해 구별되는 앞에 있는 행 수에 1을 더해서 정의됩니다. 그러므로, 순차 순위 지정에 간격이 없습니다.

ROW_NUMBER[37] 함수는 첫번째 행을 1로 시작하여, 창 내에서 순서 지정에 의해 정의된 행에 대한 순차 행 번호를 계산합니다. ORDER BY 절이 창에 지정되지 않을 경우, 행 번호는 부속 선택에 의해 리턴되는 임시 순서로 행에 지정됩니다(선택 명령문의 ORDER BY 절에 따르지 않고).

RANK, DENSE_RANK 또는 ROW_NUMBER의 결과에 대한 데이터 유형은 BIGINT입니다. 결과는 널(NULL)이 될 수 없습니다.

PARTITION BY (*파티션 표현식*,...)

함수가 적용되는 파티션을 정의합니다. *파티션 표현식*은 결과 세트의 파티션을 정의하는 데 사용되는 표현식입니다. 파티션 표현식에서 참조되는 각 *컬럼 이름*은 OLAP 함수 부속 선택 명령문의 결과 세트 컬럼을 확실하게 참조해야 합니다(SQLSTATE 42702 또는 42703). 각 파티션 표현식의 길이는 255 바이트보다 클 수 없습니다(SQLSTATE 42907). 파티션 표현식에는 스칼라 fullselect(SQLSTATE 42822)나, 결정할 수 없거나 외부 조치가 있는 함수(SQLSTATE 42845)는 포함할 수 없습니다.

ORDER BY (*정렬 키 표현식*,...)

OLAP 함수의 값이나 창 총계 그룹 절에서 ROW 값의 의미를 판별하는 행 순서를 파티션 내에서 정의합니다(조회 결과 세트의 순서를 정의하지 않습니다). *정렬 키 표현식*은 창 파티션 내에서 행 순서를 정의하는 데 사용되는 표현식입니다. 정렬 키 표현식에서 참조되는 각 컬럼 이름은 OLAP 함수를 포함하

36. DENSE_RANK 및 DENSERANK는 동의어입니다.

37. ROW_NUMBER와 ROWNUMBER는 동의어입니다

여 부속 선택의 결과 세트 컬럼을 확실하게 참조해야 합니다(SQLSTATE 42702 또는 42703). 각 정렬 키 표현식의 길이 속성은 255 바이트보다 클 수 없습니다(SQLSTATE 42907). 정렬 키 표현식에는 스칼라 fullselect (SQLSTATE 42822)나, 결정할 수 없거나 외부 조치가 있는 함수(SQLSTATE 42845)는 포함할 수 없습니다. 이 절은 RANK 및 DENSE_RANK 함수에 대해 반드시 필요합니다(SQLSTATE 42601).

ASC

정렬 키 표현식의 값을 오름차순으로 사용합니다. 널(NULL) 값은 순서에서 마지막으로 고려됩니다.

DESC

정렬 키 표현식의 값을 내림차순으로 사용합니다. 널(NULL) 값은 순서에서 첫번째로 고려됩니다.

창 총계 그룹 절

행 R의 총계 그룹은 R 파티션 행들의 순서 지정에서 R에 상대적으로 정의된 행 세트입니다. 이 절은 총계 그룹을 지정합니다.

ROWS

총계 그룹이 계산 행에 의해 정의됨을 나타냅니다.

RANGE

총계 그룹이 정렬 키로부터의 오프셋에 의해 정의됨을 나타냅니다.

그룹 시작

총계 그룹에 대한 시작점을 지정합니다. 총계 그룹 끝은 현재 행입니다. 그룹 시작 절의 스펙은 양식 ″BETWEEN group-start AND CURRENT ROW″의 그룹 between 절과 같습니다.

그룹 between

ROWS 또는 RANGE를 기초로 총계 그룹 시작 및 끝을 지정합니다.

UNBOUNDED PRECEDING

현재 행 앞에 있는 전체 파티션을 포함합니다. 이것은 ROWS 또는 RANGE로 지정될 수 있습니다. 또한, 이것은 창 순서 절에서 다중 정렬 키 표현식으로 지정할 수 있습니다.

UNBOUNDED FOLLOWING

현재 행 다음에 있는 전체 파티션을 포함합니다. 이것은 ROWS 또는 RANGE로 지정될 수 있습니다. 또한, 이것은 창 순서 절에서 다중 정렬 키 표현식으로 지정할 수 있습니다.

CURRENT ROW

총계 그룹의 시작과 끝은 현재 행으로 지정합니다. 이 절은 *그룹 경계1*이 *값* FOLLOWING을 지정할 경우에 지정할 수 없습니다.

값 **PRECEDING**

현재 행 앞에 있는 행 범위나 행 수를 지정합니다. ROWS를 지정할 경우, *값*은 행 수를 나타내는 양의 정수입니다. RANGE를 지정할 경우, *값*의 데이터 유형은 창 순서 절의 정렬 키 표현식 유형에 비교할 수 있어야 합니다. 하나의 정렬 키 표현식이 있을 수 있으며 정렬 키 표현식의 데이터 유형은 빼기를 허용해야 합니다. 이 절은 *그룹 경계1*이 CURRENT ROW이거나 *값* FOLLOWING일 경우에는 *그룹 경계2*에 지정할 수 없습니다.

값 **FOLLOWING**

현재 행 다음에 있는 행 범위나 행 수를 지정합니다. ROWS를 지정할 경우, *값*은 행 수를 나타내는 양의 정수입니다. RANGE를 지정할 경우, *값*의 데이터 유형은 창 순서 절의 정렬 키 표현식 유형에 비교할 수 있어야 합니다. 하나의 정렬 키 표현식이 있을 수 있으며 정렬 키 표현식의 데이터 유형은 더하기를 허용해야 합니다.

예:

- $30,000을 초과하는 총 급료를 받는 사원들의 순위를, 총 급료에 따라(급료에 보너스를 더한 값을 기초로) 성 순서로 표시합니다.

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
       RANK() OVER (ORDER BY SALARY+BONUS  DESC) AS RANK_SALARY
  FROM EMPLOYEE WHERE SALARY+BONUS > 30000
  ORDER BY LASTNAME
```

결과 순서가 순위에 지정될 경우, ORDER BY LASTNAME을 다음으로 대체합니다.

```
ORDER BY RANK_SALARY
```

또는

```
ORDER BY RANK() OVER (ORDER BY SALARY+BONUS DESC)
```

- 평균 총 급료에 따라 부서 순위를 지정합니다.

```
SELECT WORKDEPT, AVG(SALARY+BONUS) AS AVG_TOTAL_SALARY,
       RANK() OVER (ORDER BY AVG(SALARY+BONUS ) DESC) AS RANK_AVG_SAL
   FROM EMPLOYEE
   GROUP BY WORKDEPT
   ORDER BY RANK_AVG_SAL
```

- 교육 수준에 따라 부서 내에서 사원들의 순위를 지정합니다. 부서에서 여러 사원들의 순위가 같으면 다음 순위 값을 증가시키지 말아야 합니다.

```
SELECT WORKDEPT, EMPNO, LASTNAME, FIRSTNME, EDLEVEL
       DENSE_RANK() OVER
         (PARTITION BY WORKDEPT ORDER BY EDLEVEL DESC) AS RANK_EDLEVEL
   FROM EMPLOYEE
   ORDER BY WORKDEPT, LASTNAME
```

- 조회 결과에서 행 번호를 제공합니다.

```
SELECT ROW_NUMBER() OVER (ORDER BY WORKDEPT, LASTNAME) AS NUMBER,
       LASTNAME, SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

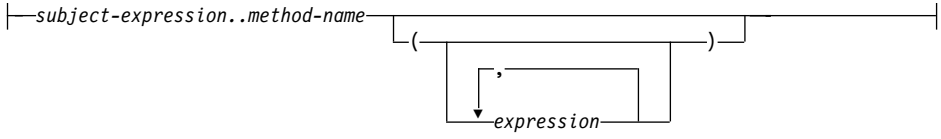
- 급료 순위 상위 5명의 사원들을 나열합니다.

```
SELECT EMPNO, LASTNAME, FIRSTNME, TOTAL_SALARY, RANK_SALARY
   FROM (SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
                RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
                FROM EMPLOYEE) AS RANKED_EMPLOYEE
   WHERE RANK_SALARY < 6
   ORDER BY RANK_SALARY
```

WHERE 절에서 순위가 사용되기 전에 결과(순위를 포함하여)를 처음 계산할 때 중첩 테이블 표현식이 사용되었습니다. 공통 테이블 표현식도 사용될 수 있었습니다.

메소드 호출

method-invocation:



사용자 정의 메소드와, 시스템에서 생성되는 observer 및 mutator 메소드는 이중 점 연산자를 사용하여 호출됩니다.

주제 표현식

사용자 정의 구조화 유형인 정적 결과 유형을 가지고 있는 표현식.

메소드 이름

메소드의 규정되지 않은 이름. *주제 표현식*의 정적 유형이나 해당되는 수퍼 유형 중 하나에 지정된 이름의 메소드가 포함되어야 합니다.

(*표현식***,...)**

*메소드 이름*의 인수는 괄호 안에 지정됩니다. 빈 괄호는 인수가 없음을 나타내기 위해 사용될 수 있습니다. *메소드 이름*과 지정된 인수 표현식의 데이터 유형은 *주제 표현식*의 정적 유형을 기초로 특정 메소드로 분석됩니다(176 페이지의 『메소드 분석』에서 자세한 내용을 참조하십시오).

메소드 호출에 사용되는 이중 점 연산자는 왼쪽에서 오른쪽으로 삽입되는 상위 우선순위 연산자입니다. 예를 들어, 다음의 두 표현식은 같습니다.

```
a..b..c + x..y..z
```

와

```
((a..b)..c) + ((x..y)..z)
```

메소드에 주제 이외에 매개변수가 없을 경우, 괄호를 사용하거나 괄호 없이 호출될 수 있습니다. 예를 들어, 다음의 두 표현식은 같습니다.

```
point1..x
```

와

```
point1..x()
```

메소드 호출에서의 널(Null) 주제는 다음과 같이 처리됩니다.

1. 시스템 생성 mutator 메소드가 널(NULL) 주제로 호출될 경우, 오류가 발생합니다(SQLSTATE 2202D).
2. 시스템 생성 mutator 이외의 메소드가 널(NULL) 주제로 호출될 경우, 메소드는 실행되지 않고 그 결과는 널(NULL)이 됩니다. 이 규칙에는 SELF AS RESULT와 함께 사용자 정의 메소드가 포함됩니다.

데이터베이스 오브젝트(예: 패키지, 뷰 또는 트리거)가 작성될 경우, 메소드 호출 각각에 대해 존재하는 최적의 메소드는 176 페이지의 『메소드 분석』에 지정된 규칙을 사용하여 찾을 수 있습니다.

주:

- WITH FUNCTION ACCESS에서 정의된 메소드 유형들은 일반 함수 표기로도 호출할 수 있습니다. 함수 결정에서는 함수 액세스를 가지고 있는 메소드와 모든 함수를 후보 함수로 고려합니다. 그러나, 함수들은 메소드 호출로 호출할 수 없습니다. 메소드 결정에서는 모든 메소드를 고려하지만 함수를 후보 메소드로 고려하지는 않습니다. 적절한 함수나 메소드에 대한 결정에 실패하면 오류가 발생합니다(SQLSTATE 42884).

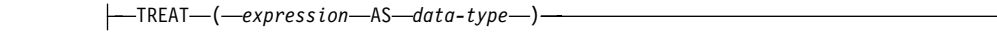
예:

- AREA라고 하는 메소드를 호출하기 위해 이중 점 연산자를 사용합니다. 구조화 유형 CIRCLE의 컬럼 CIRCLE_COL이 있는 RINGS라고 하는 테이블이 있다고 가정합시다. 또한, 메소드 AREA가 이전에 CIRCLE 유형에 대해 AREA() RETURNS DOUBLE로 정의되었다고 가정합시다.

```
SELECT CIRCLE_COL..AREA()
   FROM RINGS
```

부속 유형 처리

subtype-treatment:



*부속 유형 처리*는 구조화 유형 표현식을 부속 유형 중 하나로 유형변환하는 데 사용됩니다. *표현식*의 정적 유형은 사용자가 정의하는 구조화 유형이어야 하고, 그 유형은 *데이터 유형*과 같거나 이 유형의 수퍼 유형이어야 합니다. *데이터 유형*의 유형 이름이 규정되지 않을 경우, 유형 참조를 분석하기 위해 SQL 경로가 사용됩니다. 부속 유형 처리의 결과에 대한 정적 유형은 *데이터 유형*이고, 부속 유형 처리의 값은 표현식의 값입니다. 런타임에서, 표현식의 동적 유형이 *데이터 유형*이 아니거나 *데이터 유형*의 부속 유형이 아니면, 오류가 발생합니다(SQLSTATE 0D000).

예:

- 컬럼 CIRCLE_COL의 모든 컬럼 오브젝트 인스턴스가 동적 유형 COLOREDCIRCLE을 가지고 있다는 것을 응용프로그램이 인식할 경우, 다음 조회를 사용하여 그러한 오브젝트에서 메소드 RGB를 호출합니다. 구조화 유형 CIRCLE의 컬럼 CIRCLE_COL이 있는 RINGS라고 하는 테이블이 있다고 가정합시다. 또한, COLOREDCIRCLE이 CIRCLE의 부속 유형이고 메소드 RGB가 이전에 COLOREDCIRCLE에 대해 RGB() RETURNS DOUBLE로 정의되었다고 가정합시다.

  ```
  SELECT TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
    FROM RINGS
  ```

  런타임에서, 동적 유형 CIRCLE의 인스턴스가 있을 경우, 오류가 발생합니다 (SQLSTATE 0D000). 이 오류는 다음과 같이 CASE 표현식에서 TYPE 술어를 사용하여 피할 수 있습니다.

  ```
  SELECT (CASE
    WHEN CIRCLE_COL IS OF (COLOREDCIRCLE)
      THEN TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
      ELSE NULL
    END)
    FROM RINGS
  ```

  234 페이지의 『TYPE 술어』에서 자세한 설명을 참조하십시오.

술어

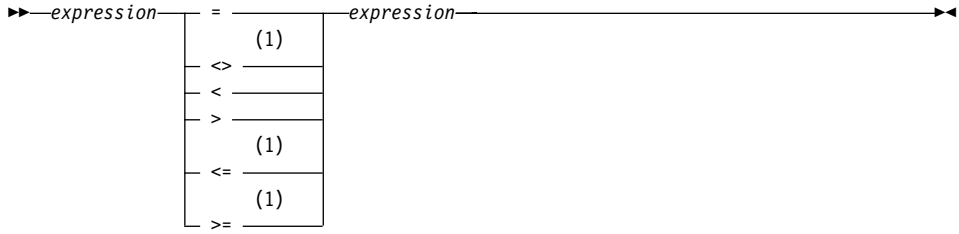
술어는 참 또는 거짓 또는 주어진 행이나 그룹에 대해 알 수 없음의 조건을 지정합니다.

다음 규칙은 술어의 모든 유형에 적용됩니다.

- 술어에 지정된 모든 값은 호환되어야 합니다.
- Basic, Quantified, IN 또는 BETWEEN 술어에 사용된 표현식은 길이 속성이 4 000보다 큰 문자열, 2 000보다 큰 그래픽 문자열 또는 임의 *크기*의 LOB 문자열이 되어서는 안 됩니다.
- 호스트 변수의 값은 널(NULL)이 될 수 있습니다(즉, 변수는 음수 표시기 값을 가질 수 있습니다.).
- LIKE를 제외한 둘 이상의 피연산자를 포함하는 술어의 피연산자 코드 페이지 변환은 129 페이지의 『문자열 변환 규칙』에 따라 수행됩니다.
- DATALINK 값의 사용은 NULL 술어에 제한됩니다.
- 구조화 유형 값을 사용할 경우 NULL 술어와 TYPE 술어로 제한됩니다.

fullselect는 433 페이지의 『제5장 조회』에서 설명한 SELECT문의 양식입니다. 술어에 사용된 fullselect를 *부속 조회(subquery)*라고도 합니다.

기본 술어



주:

1 다른 비교 연산자도 지원됩니다. [38]

*기본 술어(basic predicate)*는 값을 비교합니다.

각 피연산자의 값이 널(NULL)인 경우, 술어의 결과는 알 수 없습니다. 그렇지 않은 경우, 결과는 참이나 거짓입니다.

*x*와 *y*의 경우

술어	다음을 만족할 경우(에만) 참...
x **=** *y*	*x*가 *y*와 같습니다.
x **<>** *y*	*x*가 *y*와 같지 않습니다.
x **<** *y*	*x*가 *y*보다 작습니다.
x **>** *y*	*x*가 *y*보다 큽니다.
x **>=** *y*	*x*가 *y*보다 크거나 같습니다.
x **<=** *y*	*x*가 *y*보다 작거나 같습니다.

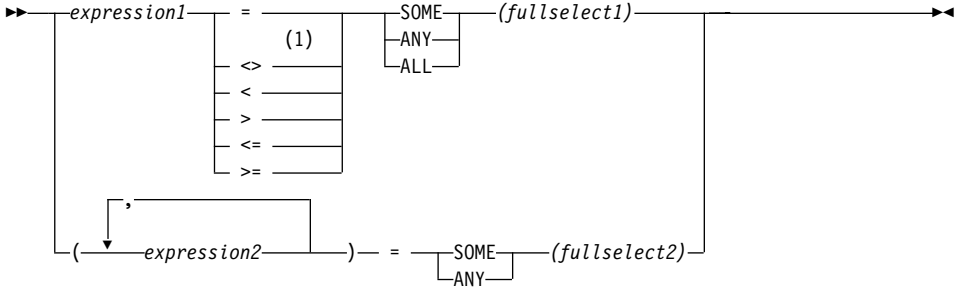
예:

```
EMPNO='528671'
SALARY < 20000
PRSTAFF <> :VAR1
SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE)
```

38. 비교 연산자의 다음 양식은 기본 및 규정된 술어 ; ^=, ^<, ^>, !=, !<, !>에서도 지원됩니다. 또한, 코드 페이지 437, 819, 850에서, 양식 ¬=, ¬<, ¬>가 지원됩니다.

    이러한 제품 고유의 비교 연산자 양식은 이러한 연산자를 사용하는 기존의 SQL만을 지원하기 위한 것으로, 새로운 SQL문을 작성할 때는 사용하지 않는 것이 좋습니다.

정량 술어



주:

1 다른 비교 연산자도 지원됩니다. [38]

*정량 술어*는 하나 또는 여러 개의 값을 값의 콜렉션과 비교합니다.

fullselect는 술어 연산자의 왼쪽에 지정된 표현식의 수와 동일한 컬럼 수를 나타내야 합니다(SQLSTATE 428C4). fullselect가 임의의 행 수를 리턴할 수 있습니다.

ALL이 지정된 경우:

- fullselect가 아무 값도 리턴하지 않거나, 지정된 관계가 fullselect가 리턴한 모든 값에 대해 참인 경우 술어 결과는 참입니다.
- 지정된 관계가 fullselect가 리턴한 값 중 적어도 하나에 대해 거짓인 경우 결과는 거짓입니다.
- 지정된 관계가 fullselect가 리턴한 값에 대해 거짓이 아니고, 적어도 하나의 비교가 널(NULL)값으로 인해 알 수 없는 경우, 결과는 알 수 없습니다.

SOME 또는 ANY가 지정된 경우:

- fullselect가 리턴한 적어도 한 행의 각 값에 대해, 지정된 관계가 참일 경우 술어의 결과는 참입니다.
- fullselect가 행을 전혀 리턴하지 않거나, fullselect가 리턴한 모든 행의 적어도 한 값에 대해 지정된 관계가 거짓일 경우, 결과는 거짓입니다.
- 지정된 관계가 임의의 행에 대해 참이 아니고, 적어도 하나의 비교가 널(NULL) 값으로 인해 알 수 없는 경우, 결과는 알 수 없습니다.

예: 다음 예를 참조할 경우 다음 테이블을 사용하십시오.

TBL***AB***:

COL***A***	COL***B***
1	12
2	12
3	13
4	14
-	-

TBL***XY***:

COL***X***	COL***Y***
2	22
3	23

그림 10.

예 1

```
SELECT COLA FROM TBLAB
   WHERE COLA = ANY(SELECT COLX FROM TBLXY)
```

2,3의 결과. 부속 선택은 (2,3)을 리턴합니다. 행2와 행3에 있는 COLA는 이러한 값 중 적어도 하나와 같습니다.

예 2:

```
SELECT COLA FROM TBLAB
   WHERE COLA > ANY(SELECT COLX FROM TBLXY)
```

3,4의 결과. 부속 선택은 (2,3)을 리턴합니다. 행3과 행4에 있는 COLA는 이러한 값중 적어도 하나보다 큽니다.

예 3:

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY)
```

4에서의 결과. 부속 선택은 (2,3)을 리턴합니다. 행 4에 있는 COLA는 이러한 값 모두보다 큰 값입니다.

예 4:

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY
                    WHERE COLX<0)
```

1,2,3,4, null에서의 결과. 부속 선택은 아무값도 리턴하지 않습니다. 그러므로 술어는 TBLAB에 있는 모든 행에 대해 참입니다.

예 5

```
SELECT * FROM TBLAB
   WHERE(COLA,COLB+10) = SOME(SELECT COLX, COLY FROM TBLXY)
```

부속 선택은 TBLXY로부터 모든 항목을 리턴합니다. 술어는 부속 선택에 대해서 참이지만, 결과는 다음과 같습니다.

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

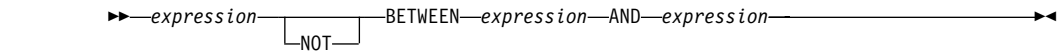
예 6

```
SELECT * FROM TBLAB
   WHERE(COLA,COLB) = ANY(SELECT COLX,COLY-10 FROM TBLXY)
```

부속 선택은 TBLXY로부터 COLX와 COLY-10을 리턴합니다. 술어는 부속 선택에 대해서 참이지만, 결과는 다음과 같습니다.

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

BETWEEN 술어



BETWEEN 술어는 어떤 범위에 있는 값을 비교합니다.

BETWEEN 술어:

```
값1 BETWEEN 값2 AND 값3
```

이는 다음의 검색 조건과 같습니다.

```
값1 >= 값2 AND 값1 <= 값3
```

BETWEEN 술어:

```
값1 NOT BETWEEN 값2 AND 값3
```

이는 다음의 검색 조건과 같습니다.

```
NOT(값1 BETWEEN 값2 AND 값3); 즉,
값1 < 값2 OR 값1 > 값3.
```

BETWEEN 술어에 있는 표현식의 값은 다른 코드 페이지를 가질 수 있습니다. 피연산자는 위와 동일한 검색 조건이 지정된 것처럼 변환됩니다.

첫번째 피연산자(표현식)에는 변형 함수가 되거나 또는 외부 조치(SQLSTATE 426804)를 갖는 함수는 포함될 수 없습니다.

날짜 시간값과 날짜 시간값의 문자열 표현이 혼합되어 있는 경우, 모든 값은 날짜 시간 피연산자의 데이터 유형으로 변환됩니다.

예:

예 1

```
EMPLOYEE.SALARY BETWEEN 20000 AND 40000
```

$20,000.00과 $40,000.00 사이의 모든 급료가 결과로 나옵니다.

예 2:

```
SALARY NOT BETWEEN 20000 + :HV1 AND 40000
```

:HV1이 5000이라고 가정하면, 결국 모든 급료는 $25,000.00 아래와 $40,000.00 위에 있게 됩니다.

예 3:

다음을 가정할 경우:

표 12.

표현식	유형	코드 페이지
HV_1	호스트-변수	437
HV_2	호스트-변수	437
Col_1	컬럼	850

술어을 평가할 경우:

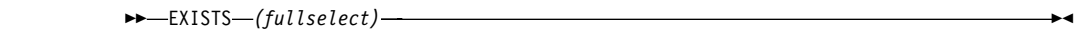
```
  :HV_1 BETWEEN :HV_2 AND COL_1
```

이것은 다음과 같이 해석됩니다.

```
  :HV_1 >= :HV_2
AND :HV_1 <= COL_1
```

:HV_1의 첫번째 발생은 응용프로그램 코드 페이지에도 남아 있게 되는 :HV_2와 비교되므로 응용프로그램 코드 페이지에 남아 있게 됩니다. :HV_1의 두 번째 발생은 컬럼 값과 비교되므로 데이터베이스 코드 페이지로 변환됩니다.

EXISTS 술어



EXISTS 술어는 일정한 행이 있는지 테스트합니다.

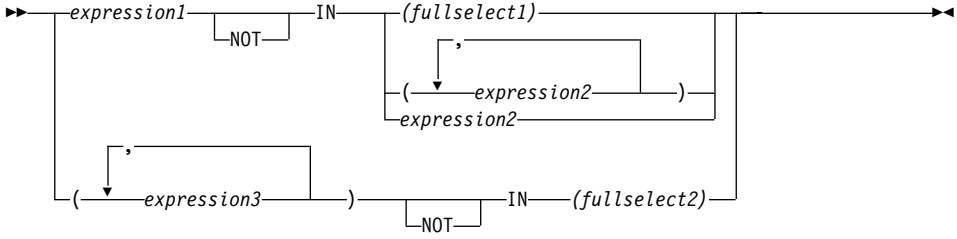
fullselect는 임의의 수의 컬럼을 지정할 수 있으며,

- fullselect가 지정한 행 수가 0이 아닌 경우에만 결과가 참입니다.
- 지정된 행수가 0인 경우에만 결과가 거짓입니다.
- 결과는 알 수 없음이 될 수 없습니다.

예:

```
EXISTS (SELECT * FROM TEMPL WHERE SALARY < 10000)
```

IN 술어



IN 술어는 하나 또는 여러 개의 값을 값의 콜렉션과 비교합니다.

fullselect는 IN 키워드의 왼쪽에 지정된 표현식의 수와 동일한 컬럼 수를 나타내야 합니다(SQLSTATE 428C4). fullselect가 임의의 행 수를 리턴할 수 있습니다.

- 다음 양식에서 IN 술어는

  표현식 **IN** 표현식

  다음의 기본 술어 양식과 같습니다.

  표현식 = 표현식

- 다음 양식에서 IN 술어는

  표현식 **IN**(fullselect)

  다음의 정량 술어 양식과 같습니다.

  표현식 = **ANY**(fullselect)

- 다음 양식에서 IN 술어는

  표현식 **NOT IN**(fullselect)

  다음의 정량 술어 양식과 같습니다.

  표현식 <> **ALL**(fullselect)

- 다음 양식에서 IN 술어는

  표현식 **IN**(표현식a, 표현식b, ..., 표현식k)

  아래와 동일합니다.

  표현식 = **ANY**(fullselect)

여기서, values절 양식은 다음과 같습니다.

```
VALUES(표현식a),(표현식b), ...,(표현식k)
```

- 다음 양식에서 IN 술어는

```
(표현식a, 표현식b,..., 표현식k) IN(fullselect)
```

다음의 정량 술어 양식과 같습니다.

```
(표현식a, 표현식b,..., 표현식k) = ANY(fullselect)
```

*표현식1*과 *표현식2*에 대한 값이나, IN 술어에 있는 *fullselect1*의 컬럼은 호환 가능해야 합니다. *표현식3* 값과 이에 해당하는 IN 술어의 *fullselect2* 컬럼은 호환 가능해야 합니다. 125 페이지의 『결과 데이터 유형 규칙』을 이용하여 비교시 사용되는 결과의 속성을 결정할 수 있습니다.

IN 술어에 있는 표현식에 대한 값(상응하는 fullselect의 컬럼 포함)은 서로 다른 코드 페이지를 가질 수 있습니다. 변환이 필요한 경우, 129 페이지의 『문자열 변환 규칙』을 IN 목록에 먼저 적용한 후, 두 번째 피연산자로 IN 목록에서 도출된 코드 페이지를 이용하는 술어에 적용하여 코드 페이지를 결정합니다.

예:

예 1: 다음은 DEPTNO 컬럼에서 평가하는 행의 값에 D01, B01 또는 C01이 있을 경우 참으로 평가됩니다.

```
DEPTNO IN('D01', 'B01', 'C01')
```

예 2: 다음은 왼쪽에 있는 EMPNO(사원 번호)가 부서 E11의 사원에 대한 EMPNO와 일치할 경우에만 참으로 평가됩니다.

```
EMPNO IN(SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')
```

예 3: 다음 정보를 가정할 경우, 이 예는 COL_1 컬럼의 행에 있는 특정 값이 목록에 있는 값과 일치할 경우 참으로 평가됩니다.

표 13. IN 술어 예

표현식	유형	코드 페이지
COL_1	컬럼	850
HV_2	호스트-변수	437
HV_3	호스트-변수	437
CON_1	상수	850

술어을 평가할 경우:

```
COL_1 IN (:HV_2, :HV_3, CON_4)
```

두 개의 호스트 변수는 129 페이지의 『문자열 변환 규칙』을 기준으로 하여 코드 페이지 850으로 변환됩니다.

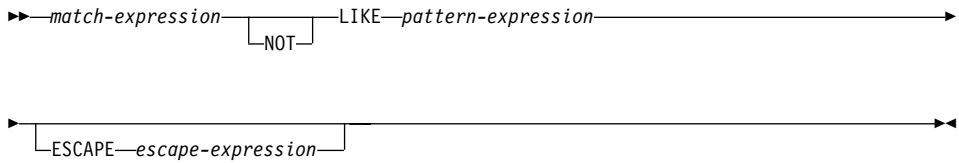
예 4: 다음은 EMENDATE(프로젝트에 대한 사원 활동이 종료된 날짜)에 지정된 연도가 목록에 지정된 값(현재 연도나 2년 전)과 일치할 경우에 참으로 평가됩니다.

```
YEAR(EMENDATE) IN(YEAR(CURRENT DATE),
                  YEAR(CURRENT DATE - 1 YEAR),
                  YEAR(CURRENT DATE - 2 YEARS))
```

예 5: 다음은 왼쪽에 있는 ID와 DEPT가 ORG 테이블의 행에 대해 각각 MANAGER 및 DEPTNUMB와 일치할 경우에 참으로 평가됩니다.

```
(ID, DEPT) IN(SELECT MANAGER, DEPTNUMB FROM ORG)
```

LIKE 술어



LIKE 술어는 일정한 패턴(pattern)을 가지고 있는 문자열을 검색합니다. 패턴은 밑줄 및 퍼센트 기호가 특수한 의미를 가질 수 있는 문자열로 지정됩니다. 패턴의 뒤에 있는 공백도 패턴의 일부입니다.

인수값이 널(NULL)인 경우, LIKE 술어의 결과는 알 수 없습니다.

일치-표현식, *패턴-표현식* 및 *escape-표현식*의 값은 호환 가능한 문자열 표현식입니다. 각 인수에서 지원하는 문자열 표현식 유형에는 약간의 차이가 있습니다. 유효한 표현식 유형은 인수의 설명 밑에 나열됩니다.

어떤 표현식도 구별 유형을 생성할 수 없습니다. 그러나, 표현식은 구별 유형에서 소스 유형으로 변환되는 함수가 될 수는 있습니다.

일치 표현식
: 일정한 패턴의 문자열에 적합한지를 결정하도록 시험하는 문자열을 지정하는 표현식.

  표현식은 다음 중 하나로 지정할 수 있습니다.

  - 상수
  - 특수 레지스터
  - 호스트 변수(위치 지정자 변수나 파일 참조 변수를 포함하여)
  - 스칼라 함수
  - 대형 오브젝트(LOB) 위치 지정자
  - 컬럼 이름
  - 위의 설명을 병합한 표현식

패턴 표현식
: 일치시킬 문자열을 지정하는 표현식.

표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 호스트 변수
- 피연산자가 위의 사항 중 하나인 스칼라 함수
- 위의 설명을 병합한 표현식

다음과 같은 제한을 받습니다.

- 표현식의 어떤 요소도 LONG VARCHAR, CLOB, LONG VARGRAPHIC 또는 DBCLOB이 될 수 없습니다. 추가로 이것은 BLOB 파일 참조 변수가 될 수 없습니다.
- *패턴-표현식*의 실제 길이는 32 672바이트보다 클 수 없습니다.

LIKE 패턴 사용의 간단한 설명은 패턴이 *패턴 표현식*에 있는 값에 대한 확인 기준을 지정하는 데 사용되는 패턴입니다. 여기서,

- 밑줄 문자(_)는 단일 문자를 나타냅니다.
- 퍼센트 부호(%)는 0개 이상의 문자로 된 문자열을 나타냅니다.
- 그밖의 문자는 자신을 나타냅니다.

*패턴 표현식*에 밑줄이나 퍼센트 문자를 포함해야 할 경우, *escape 표현식*을 사용하여 패턴에서 밑줄이나 퍼센트 문자의 앞에 붙일 문자를 지정합니다.

LIKE 패턴 사용에 대한 자세한 설명은 다음과 같습니다. 이 설명에서는 *escape-표현식*의 사용을 무시하고, 이것의 사용에 대해서는 이후에 설명하기로 합니다.

- *m*은 *일치-표현식*의 값을, *p*는 *패턴-표현식*의 값을 나타낸다고 가정합니다. *p* 문자열은 부속문자열 지정자의 최소 개수로서 해석되어 *p*의 각 문자는 한 부속문자열 지정자의 부분이 됩니다. 부속문자열 지정자는 밑줄, 퍼센트 기호 또는 그 이외의 문자열입니다.

  술어의 결과는 *m*이나 *p*가 널(NULL)값인 경우는 알 수 없습니다. 그렇지 않은 경우, 결과는 참이나 거짓입니다. 결과는 *m*과 *p*가 빈 문자열이고 다음과 같이 *m*에서 부속 문자열로의 파티셔닝이 존재할 경우에 참입니다.

- *m*의 부속문자열은 0이나 그 이상의 연속적인 일련의 문자열로서, *m*의 각 문자는 한 부속문자열의 일부입니다.
- *n*번째 부속문자열 지정자가 밑줄인 경우, *m*의 *n*번째 부속문자열은 하나의 문자입니다.
- *n*번째 부속문자열 지정자가 퍼센트인 경우, *m*의 *n*번째 부속문자열은 0이거나 그 이상의 일련의 문자입니다.
- *n*번째 부속문자열 지정자가 밑줄도 퍼센트 기호도 아닌 경우, *m*의 *n*번째 부속문자열은 부속문자열 지정자와 같으며, 부속문자열 지정자와 길이도 같습니다.
- *m*의 부속문자열 개수는 부속문자열 지정자의 개수와 같습니다.

*p*가 빈 문자열이고 *m*이 빈 문자열이 아닌 경우, 결과는 거짓입니다. 마찬가지로, *m*이 빈 문자열이고, *p*가 빈 문자열이 아닌 경우, 결과는 거짓입니다.

술어 *m* NOT LIKE *p*는 검색 조건 NOT(*m* LIKE *p*)과 같습니다.

*escape-표현식*이 지정되면, *패턴-표현식*에는 바로 다음에 escape 문자, 밑줄 또는 퍼센트 기호가 오는 경우를 제외하고 *escape-표현식*에 의해 식별되는 escape 문자를 포함해서는 안 됩니다(SQLSTATE 22025).

*일치-표현식*이 MBCS 데이터베이스의 문자열이면, 혼합 데이터가 포함될 수 있습니다. 이 경우, 패턴에는 SBCS 및 MBCS 문자가 모두 포함될 수 있습니다. 패턴에 있는 특수 문자는 다음과 같이 해석됩니다.

- SBCS 밑줄은 하나의 SBCS 문자를 지칭합니다.
- DBCS 밑줄은 하나의 MBCS 문자를 지칭합니다.
- 퍼센트(SBCS나 DBCS)는 0 또는 그 이상의 SBCS나 MBCS 문자를 지칭합니다.

escape 표현식

이 선택적 인수는 *패턴 표현식*에서 밑줄(_)과 퍼센트(%)의 특수 의미를 수정하기 위해 사용될 문자를 지정하는 표현식입니다. 이것으로 LIKE 술어가 실제 퍼센트 문자와 밑줄 문자가 들어 있는 값을 일치시키는 데 사용할 수 있습니다.

표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 호스트 변수
- 피연산자가 위의 사항 중 하나인 스칼라 함수
- 위의 설명을 병합한 표현식

다음과 같은 제한을 받습니다.

- 표현식의 어떤 요소도 LONG VARCHAR, CLOB, LONG VARGRAPHIC 또는 DBCLOB이 될 수 없습니다. 또한, BLOB 파일 참조 변수가 될 수 없습니다.
- 표현식의 결과가 SBCS 또는 DBCS 문자 또는 정확히 1바이트가 들어 있는 2진 문자열이여야 합니다(SQLSTATE 22019).

escape 문자가 패턴 문자열에 있는 경우, 밑줄, 퍼센트 기호 또는 escape 문자는 자신의 문자형 상수 어커런스를 나타낼 수 있습니다. 물음표로 된 문자 앞에 홀수개의 escape 문자가 오는 경우, 이것은 참입니다. 그밖의 경우에는 거짓입니다.

패턴에서, 연속적인 escape 문자는 다음과 같이 처리됩니다.

- S를 그런 순서로 하고, S는 큰 일련의 연속적인 escape 문자의 일부가 아니라고 가정합니다. 또한 총 n자가 들어 있다고 생각합니다. 그러면 S에 적용되는 규칙은 n의 값에 따라 달라집니다.
  - n이 홀수인 경우, S 다음에는 밑줄이나 퍼센트 기호가 와야 합니다(SQLSTATE 22025). S와 뒤에오는 문자는 escape 문자의(n-1)/2 문자형 상수 어커런스, 그 다음에 오는 밑줄이나 퍼센트 기호의 문자형 상수 어커런스를 나타냅니다.
  - n이 짝수인 경우, S는 escape 문자의 n/2 문자형 상수 어커런스를 나타냅니다. n이 짝수인 경우와는 달리, S는 패턴을 종료시킵니다. 이것이 패턴을 종료시키지 않는 경우, 이것의 다음에는 하나의 문자가 옵니다(물론, S가 연속적인 escape 문자열의 일부가 아니라는 가정에 위배된 escape 문자는 제외됩니다.). S 다음에 밑줄이나 퍼센트 기호가 오는 경우, 그 문자에는 특별한 의미가 있습니다.

다음은 escape 문자(여기서는 백슬래쉬(\))를 연속하여 사용하는 경우 어떤 영향이 있는지 보여줍니다.

패턴 문자열	실제 패턴
\%	퍼센트 기호
\\%	백슬래쉬 다음에는 0이나 그 이상의 임의의 문자가 옵니다.
\\\%	백슬래쉬 다음에는 퍼센트 기호가 옵니다.

비교에 사용된 코드 페이지는 *일치-표현식* 값의 코드 페이지를 기준으로 합니다.

- *일치-표현식*값은 변환되지 않습니다.
- *패턴-표현식*의 코드 페이지가 *일치-표현식*의 코드 페이지와 다른 경우, *패턴-표현식*의 값은 피연산자가 FOR BIT DATA로 정의(이런 경우, 변환되지 않음)되지 않는 경우 *일치-표현식*의 코드 페이지로 변환됩니다.
- *escape-표현식*의 코드 페이지가 *일치-표현식*의 코드 페이지와 다른 경우, *escape-표현식*의 값은 피연산자가 FOR BIT DATA로 정의(이런 경우 변환되지 않음)되지 않는 한 *일치-표현식*의 코드 페이지로 변환됩니다.

예:

- PROJECT 테이블에 있는 PROJNAME 컬럼에 나타나는 문자열 'SYSTEMS'를 검색합니다.

```
SELECT PROJNAME FROM PROJECT
  WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
```

- EMPLOYEE 테이블의 FIRSTNME 컬럼에서 문자의 길이가 2이고 'J'로 시작하는 문자열을 검색합니다.

```
SELECT FIRSTNME FROM EMPLOYEE
  WHERE EMPLOYEE.FIRSTNME LIKE 'J_'
```

- EMPLOYEE 테이블의 FIRSTNME 컬럼에서 첫번째 문자가 'J'인 임의 길이의 문자열을 검색합니다.

```
SELECT FIRSTNME FROM EMPLOYEE
  WHERE EMPLOYEE.FIRSTNME LIKE 'J%'
```

- CORP_SERVERS 테이블에서, CURRENT SERVER 특수 레지스터 내의 값과 일치하는 LA_SERVERS 컬럼의 문자열을 검색합니다.

```
SELECT LA_SERVERS FROM CORP_SERVERS
  WHERE CORP_SERVERS.LA_SERVERS LIKE CURRENT SERVER
```

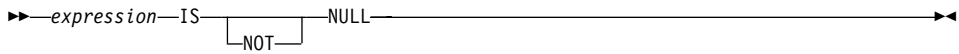
- 테이블 T의 컬럼 A에서 일련의 문자 '%_\'로 시작하는 모든 문자열을 검색합니다.

```
SELECT A FROM T WHERE T.A LIKE
'\%\_\\%' ESCAPE '\'
```

- BLOB 스칼라 함수를 사용하여 매치와 패턴 데이터 유형(BLOB 모두)과 호환되는 1바이트의 escape 문자를 구합니다.

```
SELECT COLBLOB FROM TABLET
  WHERE COLBLOB LIKE :pattern_var ESCAPE BLOB(X'0E')
```

NULL 술어



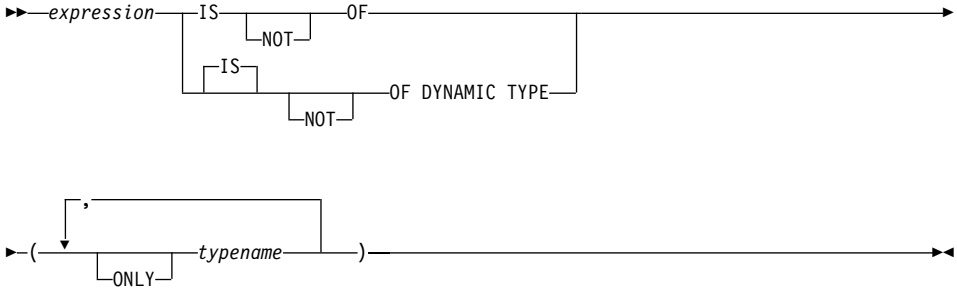
NULL 술어는 널(NULL)값에 대해 테스트합니다.

널(NULL) 술어의 결과는 알 수 없습니다. 표현식 값이 널(NULL)인 경우, 결과는 참입니다. 값이 널(NULL)이 아닌 경우, 결과는 거짓입니다. NOT이 지정된 경우, 결과는 반전됩니다.

예:

```
PHONENO IS NULL
SALARY IS NOT NULL
```

TYPE 술어



TYPE 술어는 하나 이상의 사용자 정의 구조화 유형과 표현식 유형을 비교합니다.

참조 유형의 참조 해제를 포함한 표현식의 동적 유형은 목표의 분리된 테이블 또는 뷰의 참조 행에 대한 실제 유형입니다. 이는 표현식의 정적 유형이라고 하는 참조와 관련된 표현식의 목표 유형과 다를 수 있습니다.

표현식 값이 널(NULL)인 경우, 술어 결과는 알 수 없습니다. *표현식*이 *typename*에 지정된 구조화 유형의 하위 유형중 하나이면 술어 결과는 참이며, 그렇지 않으면 거짓입니다. ONLY 앞에 *typename*이 오는 경우, 그 유형의 해당 부속 유형은 고려되지 않습니다.

*typename*이 규정화되지 않은 경우에는 SQL 경로를 사용하여 분석됩니다. 각 *typename*은 정적 *표현식* 유형의 유형 계층에 있는 사용자 정의 유형을 식별해야 합니다(SQLSTATE 428DU).

DEREF 함수는 TYPE 술어에 참조 유형 값을 포함하는 표현식이 있을 때마다 사용되어야 합니다. 이러한 *표현식* 양식에 대한 정적 유형은 참조의 목표 유형입니다. DEREF 함수에 대한 317 페이지의 『DEREF』에서 자세한 정보를 참조하십시오.

구문 IS OF 및 OF DYNAMIC TYPE은 TYPE 술어를 대신해 사용할 수 있는 것입니다. 마찬가지로, IS NOT OF 및 NOT OF DYNAMIC TYPE은 상응하는 대안입니다.

예:

테이블 계층에는 EMP 유형인 루트 테이블 EMPLOYEE와 MGR 유형인 서브테이블 MANAGER가 있습니다. 또다른 테이블 ACTIVITIES에는 REF(EMP) SCOPE EMPLOYEE로 정의된 WHO_RESPONSIBLE 컬럼이 포함됩니다. 다음은 WHO_RESPONSIBLE에 해당하는 행이 '관리자'일 경우에 참으로 평가하는 유형 술어입니다.

```
DEREF (WHO_RESPONSIBLE) IS OF (MGR)
```

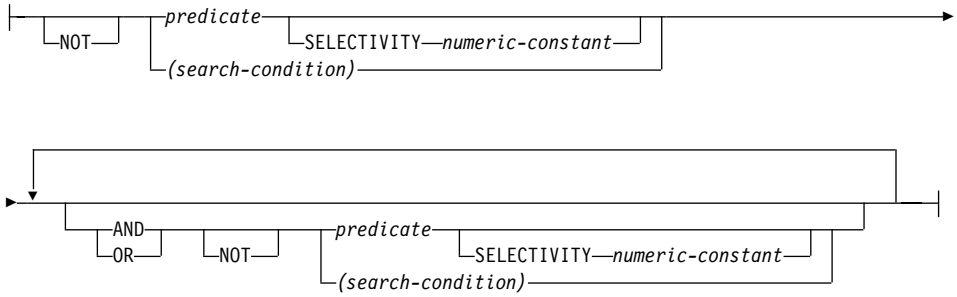
테이블에 유형이 EMP인 컬럼 EMPLOYEE가 있을 경우, EMPLOYEE에는 MGR과 같은 부속 유형의 값뿐만 아니라 유형 EMP의 값도 포함될 수 있습니다. 다음 술어는

```
EMPL IS OF (MGR)
```

EMPL이 널(NULL)이 아니고 실제로 관리 프로그램일 경우에 참을 리턴합니다.

검색 조건

search-condition:



*검색 조건(search condition)*은 조건 즉, 주어진 행에 대한 "참," "거짓" 또는 " 알 수 없음"을 지정합니다.

검색 조건의 결과는 지정된 *논리 연산자*(AND, OR, NOT)를 지정된 술어에 적용하여 얻은 것입니다. 논리 연산자가 지정되지 않은 경우, 검색 조건의 결과는 지정된 술어의 결과입니다.

AND와 OR가 표14에서 정의되고, 여기서 P와 Q는 술어입니다.

표 14. AND와 OR에 대한 참 테이블

P	Q	P AND Q	P OR Q
참	참	참	참
참	거짓	거짓	참
참	알 수 없음	알 수 없음	참
거짓	참	거짓	참
거짓	거짓	거짓	거짓
거짓	알 수 없음	거짓	알 수 없음
알 수 없음	참	알 수 없음	참
알 수 없음	거짓	거짓	알 수 없음
알 수 없음	알 수 없음	알 수 없음	알 수 없음

NOT(참)은 거짓이고, NOT(거짓)은 참이며, NOT(알 수 없음)은 알 수 없음입니다.

괄호 안에 있는 검색 조건이 먼저 평가됩니다. 평가 순서가 괄호로 지정되지 않은 경우, NOT은 AND보다 먼저 적용되고, AND는 OR보다 먼저 적용됩니다. 같은 순서 레벨에 있는 연산자가 평가되는 순서는 검색 조건의 최적화를 허용하기 위해 정의되지 않았습니다.

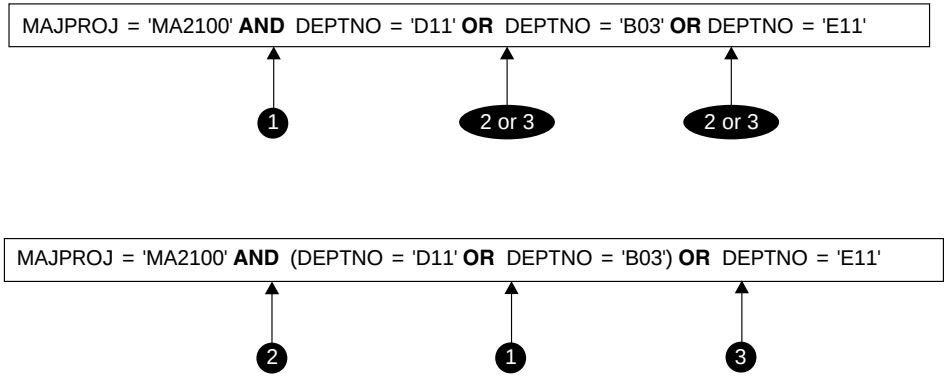


그림 11. 검색 조건 평가 순서

SELECTIVITY *값*

SELECTIVITY 절은 술어에 대해 예상되는 선택 백분율을 DB2에 표시하기 위해 사용됩니다. SELECTIVITY는 술어가 사용자 정의 술어일 경우에만 지정할 수 있습니다.

사용자 정의 술어는 CREATE FUNCTION의 PREDICATES 절에 지정한 술어 스펙과 일치하는 술어 스펙 문맥으로 되어 있는 사용자 정의 함수 호출로 구성되는 술어입니다. 예를 들어, PREDICATES WHEN=1...을 사용하여 함수 foo가 정의된 경우, 다음과 같이 SELECTIVITY를 사용하는 것은 유효합니다.

```
SELECT *
        FROM STORES
        WHERE foo(parm,parm) = 1 SELECTIVITY 0.004
```

선택성 값은 0 - 1 범위 내의 숫자 리터럴 값이어야 합니다(SQLSTATE 42615). SELECTIVITY를 지정하지 않으면, 기본값은 0.01입니다(즉, 사용자 정의 술어는 모두 필터링할 것으로 예상되지만 테이블의 모든 행에서 1 퍼센트만 필터됩니다. SELECTIVITY 기본값은 SYSSTAT.FUNCTIONS 뷰의 SELECTIVITY 컬럼을 갱신하여 제공된 함수에 맞게 변경할 수 있습니다. 사용자가 정의하지 않은 술어에 대해 SELECTIVITY 절을 지정하면 오류가 리턴됩니다(SQLSTATE 428E5).

사용자 정의 함수(UDF)는 사용자 정의 술어로 적용될 수 있으므로, 다음과 같은 경우에 잠재적으로 색인 이용에 적합합니다.

- 술어 스펙이 CREATE FUNCTION문에 존재할 경우
- 술어 스펙에 지정된 것과 같은 방법으로 비교되는(구문적으로) WHERE 절에서 UDF가 호출될 경우
- 부정(NOT 연산자)이 없을 경우

예:

다음 조회에서, WHERE 절의 within UDF 스펙은 세 개의 모든 조건을 만족하므로 사용자 정의 술어로 간주됩니다(within 및 distance UDF에 대해서는 655 페이지의 『CREATE FUNCTION(외부 스칼라)』의 예 부분을 참조하십시오.).

```
SELECT *
 FROM customers
 WHERE within(location, :sanJose) = 1 SELECTIVITY  0.2
```

그러나, 다음 조회에서의 within은 부정이 있으므로 색인 이용에 적합하지 않으므로 사용자 정의 술어로 간주되지 않습니다.

```
SELECT *
 FROM customers
 WHERE NOT(within(location, :sanJose) = 1) SELECTIVITY  0.3
```

다음 예에서, 서로 특정 거리 내에 있는 식별되는 고객과 점포를 고려해 보십시오. 점포 간의 거리는 고객이 사는 도시의 반지름으로 계산됩니다.

```
SELECT *
 FROM customers, stores
 WHERE distance(customers.loc, stores.loc)
        < CityRadius(stores.loc) SELECTIVITY 0.02
```

위의 조회에서, WHERE 절의 술어는 사용자 정의 술어로 간주됩니다. CityRadius에 의해 생성되는 결과는 범위 생성 함수에 대해 검색 인수로 사용됩니다.

그러나, CityRadius에 의해 생성되는 결과는 범위 생성 함수로 사용되므로, 위의 사용자 정의 술어는 stores.loc 컬럼에 정의된 색인 확장을 사용할 수 없게 됩니다. 그러므로, 그 UDF는 customers.loc 컬럼에 정의된 색인만 사용하게 됩니다.

제4장 함수

함수는 괄호로 묶인 인수 스펙(인수를 갖지 않을수도 있음)을 수반하며, 함수 이름에 의해 지시되는 하나의 조작입니다.

함수는 *컬럼 함수*, *스칼라 함수*, *행 함수* 또는 *테이블 함수*로 분류됩니다.

- 컬럼 함수의 인수는 유사한 값의 콜렉션입니다. 이는 하나의 값(널(NULL)이 될 수 있음)을 리턴하며 *표현식*을 사용할 수 있는 SQL 문에 지정할 수 있습니다. 258 페이지의 『컬럼 함수』에 명시된 것과 같이 컬럼 함수 사용에는 추가의 제한사항이 적용됩니다.
- 스칼라 함수의 인수는 개별적인 스칼라 값으로, 이들 값은 유형이 서로 다를 수 있으며, 의미도 서로 다릅니다. 이는 하나의 값(널(NULL)이 될 수 있음)을 리턴하며 *표현식*을 사용할 수 있는 모든 SQL문에 지정할 수 있습니다.
- 행 함수의 인수는 구조화 유형입니다. 이 함수는 내장 데이터 유형의 행을 리턴하므로 구조화 유형의 변환 함수로만 지정할 수 있습니다.
- 테이블 함수의 인수는 개별적인 스칼라 값으로, 이들 값은 유형이 서로 다를 수 있으며 의미도 서로 다릅니다. 이는 테이블을 SQL문에 리턴하며 SELECT의 FROM절 내에서만 지정할 수 있습니다. 440 페이지의 『from절』에 명시된 것과 같이 테이블 함수의 사용에는 추가의 제한사항이 적용됩니다.

240 페이지의 표15는 지원하는 함수를 보여줍니다. "스키마"와 함께 결합된 "함수 이름"은 함수의 완전 규정화된 이름을 제공합니다. "설명"은 함수가 수행하는 기능에 대해 간략하게 설명합니다. "입력 매개변수"는 함수 호출시 각 인수에 대해 기대되는 데이터 유형을 제공합니다. 많은 함수는 다른 데이터 유형이나, 다른 수의 인수들을 사용할 수 있도록 하는 입력 매개변수들의 변형을 포함합니다. 스키마, 함수 이름, 그리고 입력 매개변수들의 조합이 함수 시그니처를 구성합니다. 각 함수 시그니처는 "리턴 값" 컬럼에 표시되는 다른 유형의 값을 리턴할 수 있습니다. 입력 매개변수 유형에 대해 이해해야 하는 몇 가지의 구별사항이 있습니다. 어떤 경우에는 유형이 특정한 내장 데이터 유형으로 지정되고, 다른 경우에는 '모든 *숫자 유형*'과 같이 일반적인 변수를 사용합니다. 특정 데이터 유형이 나열되면,

이것은 지정된 데이터 유형에 대해서만 발생함을 의미합니다. 일반적인 변수가 사용되면, 그 변수와 연관되는 각 데이터 유형이 완전한 일치가 됩니다. 이러한 구별은 168 페이지의『함수 차수』에 설명된 대로 함수 선택에 영향을 줍니다.

사용자 정의 함수는 소스로 함수 시그니처 중 하나를 사용하여 서로 다른 스키마에서 작성될 수 있거나(653 페이지의『CREATE FUNCTION』참조), 사용자가 자신의 고유한 프로그램을 사용하여 외부 함수를 작성할 수 있으므로 추가로 함수를 사용할 수도 있습니다.

주:

- 내장 함수는 데이터베이스 관리 프로그램에 의해 제공되어 하나의 결과 값을 제공하며, SYSIBM 스키마의 일부로 식별됩니다. 그러한 함수의 예로는 AVG와 같은 컬럼 함수, ″+″와 같은 연산자 함수, 십진수와 같은 변환 함수 및 SUBSTR과 같은 기타 함수가 있습니다.
- 사용자 정의 함수는 SYSCAT.FUNCTIONS(CREATE FUNCTION문을 사용)에 있는 데이터베이스에 등록된 함수입니다. 사용자 정의 함수는 SYSIBM 스키마의 일부가 아닙니다. 그러한 함수 집합 중 하나가 SYSFUN 스키마에서 데이터베이스 관리 프로그램과 함께 제공됩니다.

표 15. 지원되는 함수

함수 이름	스키마	설명	
	입력 매개변수		리턴
ABS 또는 ABSVAL	SYSFUN	인수의 절대 값을 리턴합니다.	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
ACOS	SYSFUN	인수의 아크코사인을 라디안으로 표시되는 각도로 리턴합니다.	
	DOUBLE		DOUBLE
ASCII	SYSFUN	인수의 가장 왼쪽 문자의 ASCII 코드 값을 정수로 리턴합니다.	
	CHAR		INTEGER
	VARCHAR(4000)		INTEGER
	CLOB(1M)		INTEGER
ASIN	SYSFUN	인수의 아크사인을 라디안으로 표시되는 각도로 리턴합니다.	
	DOUBLE		DOUBLE

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
ATAN	SYSFUN	인수의 아크탄젠트를 라디안으로 표시되는 각도로 리턴합니다.	
	DOUBLE		DOUBLE
ATAN2	SYSFUN	각각 첫번째와 두 번째 인수로 지정되는 *x* 및 *y* 좌표를 라디안으로 표시되는 각도로 리턴합니다.	
	DOUBLE, DOUBLE		DOUBLE
AVG	SYSIBM	숫자 세트의 평균을 리턴합니다(컬럼 함수).	
	숫자 유형 [4]		*숫자 유형* [1]
BIGINT	SYSIBM	정수 상수 형식의 숫자 또는 문자열에 대해 64비트 정수 표현을 리턴합니다.	
	숫자 유형		BIGINT
	VARCHAR		BIGINT
BLOB	SYSIBM	선택적 길이로, 소스 유형으로부터 BLOB로 변환합니다.	
	문자열 유형		BLOB
	문자열 유형, INTEGER		BLOB
CEIL 또는 CEILING	SYSFUN	인수보다 크거나 같은 최소의 정수를 리턴합니다.	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
CHAR	SYSIBM	소스 유형의 문자열 표현을 리턴합니다.	
	문자 유형		CHAR
	문자 유형, INTEGER		CHAR(*정수*)
	날짜 시간 유형		CHAR
	날짜시간 유형, *키워드* [2]		CHAR
	SMALLINT		CHAR(6)
	INTEGER		CHAR(11)
	BIGINT		CHAR(20)
	DECIMAL		CHAR(2+*precision*)
	DECIMAL, VARCHAR		CHAR(2+*precision*)
CHAR	SYSFUN	부동 소수의 문자열 표현을 리턴합니다.	
	DOUBLE		CHAR(24)
CHR	SYSFUN	인수로 지정된 ASCII 코드 값을 갖고 있는 문자를 리턴합니다. 인수 값은 0과 255 사이어야 합니다. 그렇지 않으면, 리턴 값은 널(NULL)입니다.	
	INTEGER		CHAR(1)
CLOB	SYSIBM	선택적 길이로, 소스 유형으로부터 CLOB로 변환합니다.	
	문자 유형		CLOB
	문자 유형, INTEGER		CLOB

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
COALESCE [3]	SYSIBM	인수의 세트에서 첫번째 널(NULL)이 아닌 인수를 리턴합니다.	
	모든 유형, 호환가능한 모든 union, ...		*모든 유형*
CONCAT 또는 ||	SYSIBM	2개의 문자열의 병합을 리턴합니다.	
	문자열 유형, 호환가능 문자열 유형		*최대 문자열 유형*
CORRELATION 또는 CORR	SYSIBM	수 쌍 집합 상관의 계수를 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBLE
COS	SYSFUN	인수의 코사인을 리턴하며, 여기서 인수는 라디안으로 표시되는 각도입니다.	
	DOUBLE		DOUBLE
COT	SYSFUN	인수(argument)의 코탄젠트를 리턴하며, 여기서 인수는 라디안으로 표시되는 각도입니다.	
	DOUBLE		DOUBLE
COUNT	SYSIBM	행이나 값 집합에 있는 행 수의 계수를 리턴합니다(컬럼 함수).	
	모든 내장 유형 [4]		INTEGER
COUNT_BIG	SYSIBM	행이나 값 집합 내내의 행의 수나 값의 수를 리턴합니다(컬럼 함수). 결과는 정수의 최대 값보다 클 수도 있습니다.	
	모든 내장 유형 [4]		DECIMAL(31,0)
COVARIANCE 또는 COVAR	SYSIBM	수 쌍의 집합의 편차를 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBLE
DATE	SYSIBM	단일 입력 값으로부터 날짜를 리턴합니다.	
	DATE		DATE
	TIMESTAMP		DATE
	DOUBLE		DATE
	VARCHAR		DATE
DAY	SYSIBM	값의 일(day) 부분을 리턴합니다.	
	VARCHAR		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
DAYNAME	SYSFUN	db2start가 발행되었을 때의 로케일에 따라 인수의 요일 부분에 대해 요일의 이름(예: 금요일)을 포함하는 혼합 문자열을 리턴합니다.	
	VARCHAR(26)		VARCHAR(100)
	DATE		VARCHAR(100)
	TIMESTAMP		VARCHAR(100)

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
DAYOFWEEK	SYSFUN	인수에 있는 요일을 1-7 범위내의 정수 값으로 리턴합니다. 여기서 1은 일요일을 나타냅니다.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
DAYOFWEEK_ISO	SYSFUN	인수에 있는 요일을 1-7 범위내의 정수 값으로 리턴합니다. 여기서 1은 월요일을 나타냅니다.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
DAYOFYEAR	SYSFUN	인수에 있는 연도의 날짜를 1-366 범위내의 정수 값으로 리턴합니다.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
DAYS	SYSIBM	날짜의 정수 표현을 리턴합니다.	
	VARCHAR		INTEGER
	TIMESTAMP		INTEGER
	DATE		INTEGER
DBCLOB	SYSIBM	선택적 길이로, 소스 유형으로부터 DBCLOB로 변환합니다.	
	그래픽 유형		DBCLOB
	그래픽 유형, INTEGER		DBCLOB
DECIMAL or DEC	SYSIBM	선택적 정밀도와 스케일로, 숫자의 십진 표현을 리턴합니다.	
	숫자 유형		DECIMAL
	숫자 유형, INTEGER		DECIMAL
	숫자 유형 INTEGER, INTEGER		DECIMAL
DECIMAL or DEC	SYSIBM	선택적 정밀도, 스케일 및 십진 문자로 문자열의 십진 표현을 리턴합니다.	
	VARCHAR		DECIMAL
	VARCHAR, INTEGER		DECIMAL
	VARCHAR, INTEGER, INTEGER		DECIMAL
	VARCHAR, INTEGER, INTEGER, VARCHAR		DECIMAL
DEGREES	SYSFUN	라디안으로 표시된 인수에서 변환된 차수(degree)의 수를 리턴합니다.	
	DOUBLE		DOUBLE
DEREF	SYSIBM	참조 유형 인수의 목표 유형 인스턴스를 리턴합니다.	
	정의된 영역으로 된 REF(*구조화 유형*)		*구조화 유형(입력 목표 유형과 동일)*

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		리턴
DIFFERENCE	SYSFUN	SOUNDEX 함수를 사용하여 판별된 대로, 두 개의 인수 문자열에 있는 단어들의 사운드간 차이를 리턴합니다. 4의 값은 문자열 사운드가 같음을 의미합니다.	
	VARCHAR(4000), VARCHAR(4000)		INTEGER
DIGITS	SYSIBM	숫자의 문자열 표현을 리턴합니다.	
	DECIMAL		CHAR
DLCOMMENT	SYSIBM	데이터 링크 값의 주석 속성을 리턴합니다.	
	DATALINK		VARCHAR(254)
DLLINKTYPE	SYSIBM	데이터 링크 값의 링크 유형 속성을 리턴합니다.	
	DATALINK		VARCHAR(4)
DLURLCOMPLETE	SYSIBM	데이터 링크 값의 완전한 URL(액세스 토큰 포함)을 리턴합니다.	
	DATALINK		VARCHAR
DLURLPATH	SYSIBM	데이터 링크 값의 경로 및 파일 이름(액세스 토큰 포함)을 리턴합니다.	
	DATALINK		VARCHAR
DLURLPATHONLY	SYSIBM	데이터 링크 값의 경로 및 파일 이름(액세스 토큰 없음)을 리턴합니다.	
	DATALINK		VARCHAR
DLURLSCHEME	SYSIBM	데이터 링크 값의 URL 속성으로부터 체계를 리턴합니다.	
	DATALINK		VARCHAR
DLURLSERVER	SYSIBM	데이터 링크 값의 URL 속성으로부터 서버를 리턴합니다.	
	DATALINK		VARCHAR
DLVALUE	SYSIBM	데이터 위치 인수, 링크 유형 인수, 선택적 주석 문자열 인수로부터 데이터 링크 값을 구축합니다.	
	VARCHAR		DATALINK
	VARCHAR, VARCHAR		DATALINK
	VARCHAR, VARCHAR, VARCHAR		DATALINK
DOUBLE 또는 DOUBLE_PRECISION	SYSIBM	숫자의 부동 소수점 표현을 리턴합니다.	
	숫자 유형		DOUBLE
DOUBLE	SYSFUN	숫자의 문자열 표현에 해당되는 부동 소수를 리턴합니다. *인수*의 앞뒤 공백은 무시됩니다.	
	VARCHAR		DOUBLE
EVENT_MON_STATE	SYSIBM	특수한 이벤트 모니터의 작동 상태를 리턴합니다.	
	VARCHAR		INTEGER
EXP	SYSFUN	인수의 지수 함수를 리턴합니다.	
	DOUBLE		DOUBLE
FLOAT	SYSIBM	DOUBLE과 동일합니다.	

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
FLOOR	SYSFUN	인수보다 작거나 같은 최대의 정수를 리턴합니다.	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
GENERATE_UNIQUE	SYSIBM	동일 함수의 다른 실행과 비교하여 고유한 비트 날짜 문자열을 리턴합니다.	
	인수가 아님		CHAR(13) FOR BIT DATA
GRAPHIC	SYSIBM	선택적 길이로, 소스 유형으로부터 GRAPHIC으로 변환합니다.	
	그래픽 유형		GRAPHIC
	그래픽 유형, INTEGER		GRAPHIC
GROUPING	SYSIBM	그룹 세트가 생성한 소계 행을 가리키는 데 그룹 세트와 수퍼 그룹이 사용됩니다. 리턴되는 값은 다음과 같습니다. **1** 리턴된 행에서 인수의 값은 널(NULL) 값이고, 행은 그룹 세트에 대해 생성됩니다. 이 생성된 행은 그룹 세트에 대해 소계를 제공합니다. **0** 그렇지 않으면,	
	모든 유형		SMALLINT
HEX	SYSIBM	값의 16 진법 표현을 리턴합니다.	
	모든 내장 유형		VARCHAR
HOUR	SYSIBM	값의 시간을 리턴합니다.	
	VARCHAR		INTEGER
	TIME		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
INSERT	SYSFUN	*인수3* 바이트가 *인수2*에서 시작하는 *인수1*에서 삭제되고 *인수4*가 *인수2*에서 시작하는 *인수1*에 삽입된 문자열을 리턴합니다.	
	VARCHAR(4000), INTEGER, INTEGER, VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M), INTEGER, INTEGER, CLOB(1M)		CLOB(1M)
	BLOB(1M), INTEGER, INTEGER, BLOB(1M)		BLOB(1M)
INTEGER 또는 INT	SYSIBM	숫자의 정수 표현을 리턴합니다.	
	숫자 유형		INTEGER
	VARCHAR		INTEGER

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
JULIAN_DAY	SYSFUN	B.C 4712년 1월 1일(율리우스력의 시작)에서 *인수*에 지정된 날짜 값까지의 일 수를 나타내는 정수 값을 리턴합니다.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
LCASE 또는 LOWER	SYSIBM	모든 문자가 대문자로 변환된 문자열을 리턴합니다.	
	CHAR		CHAR
	VARCHAR		VARCHAR
LCASE	SYSFUN	모든 문자가 대문자로 변환된 문자열을 리턴합니다. LCASE는 불변 세트에 있는 핸들 문자만 처리합니다. 따라서, LCASE(UCASE(문자열))에서는 LCASE(문자열)와 동일한 결과를 리턴할 필요가 없습니다.	
	VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M)		CLOB(1M)
LEFT	SYSFUN	*인수1*의 가장 오른쪽 *인수2* 바이트로 구성되는 문자열을 리턴합니다.	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)
LENGTH	SYSIBM	바이트(문자 내의 길이를 리턴하는 더블 바이트 문자열 유형 제외)내의 피연산자 길이를 리턴합니다.	
	모든 내장 유형		INTEGER
LN	SUSFUN	인수의 자연 로그를 리턴합니다(LOG와 동일).	
	DOUBLE		DOUBLE
LOCATE	SYSFUN	*인수2* 내에서 *인수1*이 처음 발생한 시작 지점을 리턴합니다. 선택적인 세 번째 인수가 지정되면, *인수2*에서 검색이 시작되는 문자 위치를 나타냅니다. *인수2* 내에서 *인수1*이 발견되지 않으면, 값 0이 리턴됩니다.	
	VARCHAR(4000), VARCHAR(4000)		INTEGER
	VARCHAR(4000), VARCHAR(4000), INTEGER		INTEGER
	CLOB(1M), CLOB(1M)		INTEGER
	CLOB(1M), CLOB(1M), INTEGER		INTEGER
	BLOB(1M), BLOB(1M)		INTEGER
	BLOB(1M), BLOB(1M), INTEGER		INTEGER
LOG	SYSFUN	인수의 자연 로그를 리턴합니다(LN과 동일).	
	DOUBLE		DOUBLE
LOG10		인수의 기본 10 로그를 리턴합니다.	
	DOUBLE		DOUBLE

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명
	입력 매개변수	**리턴**
LONG_VARCHAR	SYSIBM	long 문자열을 리턴합니다.
	문자 유형	LONG VARCHAR
LONG_VARGRAPHIC	SYSIBM	소스 유형에서 LONG_VARGRAPHIC으로 변환합니다.
	그래픽 유형	LONG VARGRAPHIC
LTRIM	SYSIBM	선행 공백을 제거하고 인수의 문자를 리턴합니다.
	CHAR	VARCHAR
	VARCHAR	VARCHAR
	GRAPHIC	VARGRAPHIC
	VARGRAPHIC	VARGRAPHIC
LTRIM	SYSFUN	선행 공백을 제거하고 인수의 문자를 리턴합니다.
	VARCHAR(4000)	VARCHAR(4000)
	CLOB(1M)	CLOB(1M)
MAX	SYSIBM	값 집합의 최대 값을 리턴합니다(컬럼 함수).
	내장된 유형 [5]	*입력 유형과 동일*
MICROSECOND	SYSIBM	값의 마이크로 세컨드(시간 단위) 부분을 리턴합니다.
	VARCHAR	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
MIDNIGHT_SECONDS	SYSFUN	밤 12시와 *인수*에 지정된 시간 값 사이의 초 수를 나타내는 0 - 86 400 범위의 정수값을 리턴합니다.
	VARCHAR(26)	INTEGER
	TIME	INTEGER
	TIMESTAMP	INTEGER
MIN	SYSIBM	값 집합의 최소값을 리턴합니다(컬럼 함수).
	내장된 유형 [5]	*입력 유형과 동일*
MINUTE	SYSIBM	값의 분 부분을 리턴합니다.
	VARCHAR	INTEGER
	TIME	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
MOD	SYSFUN	*인수2*로 나눈 *인수1*의 나머지(모듈러스)를 리턴합니다. 결과는 *인수1*이 음수일 경우에만 음수입니다.
	SMALLINT, SMALLINT	SMALLINT
	INTEGER, INTEGER	INTEGER
	BIGINT, BIGINT	BIGINT

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
MONTH	SYSIBM	값의 월 부분을 리턴합니다.	
	VARCHAR		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
MONTHNAME	SYSFUN	데이터베이스가 시작되었을 당시의 로케일에 따라, 날짜 또는 시간소인 인수의 월 부분에 대한 월의 이름(예: 1월)이 들어 있는 대소문자 혼합 문자열을 리턴합니다.	
	VARCHAR(26)		VARCHAR(100)
	DATE		VARCHAR(100)
	TIMESTAMP		VARCHAR(100)
NODENUMBER [3]	SYSIBM	행의 노드 번호를 리턴합니다. 인수는 테이블내의 컬럼 이름입니다.	
	모든 유형		INTEGER
NULLIF [3]	SYSIBM	인수가 동일하면 널(NULL)을 리턴하고, 그렇지 않으면 첫번째 인수를 리턴합니다.	
	모든 유형 [5], *비교가능한 유형* [5]		*모든 유형*
PARTITION [3]	SYSIBM	행의 파티션 맵핑 색인(0 - 4095)을 리턴합니다. 인수는 테이블내의 컬럼 이름입니다.	
	모든 유형		INTEGER
POSSTR	SYSIBM	한 문자열이 또 다른 문자열에 포함된 위치를 리턴합니다.	
	문자열 유형, *호환가능 문자열 유형*		INTEGER
POWER	SYSFUN	*인수1*의 값을 *인수2*의 승수로 리턴합니다.	
	INTEGER, INTEGER		INTEGER
	BIGINT, BIGINT		BIGINT
	DOUBLE, INTEGER		DOUBLE
	DOUBLE, DOUBLE		DOUBLE
QUARTER	SYSFUN	인수에 지정된 날짜에 대해 한 해의 분기를 나타내는 1에서 4까지의 정수 값을 리턴합니다.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
RADIANS	SYSFUN	차수(degree)로 표시된 인수에서 변환된 라디안의 수를 리턴합니다.	
	DOUBLE		DOUBLE
RAISE_ERROR[3]	SYSIBM	SQLCA에서 오류를 발생시킵니다. 리턴된 sqlstate는 *인수1*에 의해 표시됩니다. 두 번째 인수에는 리턴될 텍스트가 들어 갑니다.	
	VARCHAR, VARCHAR		*모든 유형* [6]

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
RAND	SYSFUN	인수를 선택적 시드 값으로 사용하여 0에서 1 사이의 임의 부동 소수 값을 리턴합니다.	
	인수가 필요없음		DOUBLE
	INTEGER		DOUBLE
REAL	SYSIBM	숫자의 단일 정밀도 부동 소수점 표현을 리턴합니다.	
	숫자 유형		REAL
REGR_AVGX	SYSIBM	진단 통계를 계산하는 데 사용된 양을 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBLE
REGR_AVGY	SYSIBM	진단 통계를 계산하는 데 사용된 양을 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBLE
REGR_COUNT	SYSIBM	역행 라인을 맞추는 데 사용된 널(NULL)이 아닌 수 쌍의 갯수를 리턴합니다.	
	숫자 유형, 숫자 유형		INTEGER
REGR_INTERCEPT 또는 REGR_ICPT	SYSIBM	역행 라인의 y 절편을 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBLE
REGR_R2	SYSIBM	역행을 위한 결정의 계수를 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBE
REGR_SLOPE	SYSIBM	라인의 기울기를 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBLE
REGR_SXX	SYSIBM	진단 통계를 계산하는 데 사용된 양을 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBLE
REGR_SXY	SYSIBM	진단 통계를 계산하는 데 사용된 양을 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBLE
REGR_SYY	SYSIBM	진단 통계를 계산하는 데 사용된 양을 리턴합니다.	
	숫자 유형, 숫자 유형		DOUBLE
REPEAT	SYSFUN	*인수2*배 반복되는 *인수1*로 구성되는 문자열을 리턴합니다.	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)
REPLACE	SYSFUN	*인수1*에서 모든 *인수2*의 어커런스를 *인수3*으로 대체합니다.	
	VARCHAR(4000), VARCHAR(4000), VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M), CLOB(1M), CLOB(1M)		CLOB(1M)
	BLOB(1M), BLOB(1M), BLOB(1M)		BLOB(1M)
RIGHT	SYSFUN	*인수1*의 가장 오른쪽 *인수2* 바이트로 구성되는 문자열을 리턴합니다.	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
ROUND	SYSFUN	소수점의 **오른쪽** *인수2* 자리로 반올림된 첫번째 인수를 리턴합니다. *인수2*가 음수이면, *인수1*은 소수점 **왼쪽**으로 *인수2* 자리의 절대값으로 반올림됩니다.	
	INTEGER, INTEGER		INTEGER
	BIGINT, INTEGER		BIGINT
	DOUBLE, INTEGER		DOUBLE
RTRIM	SYSIBM	뒤 공백을 제거한 채 인수 문자를 리턴합니다.	
	CHAR		VARCHAR
	VARCHAR		VARCHAR
	GRAPHIC		VARGRAPHIC
	VARGRAPHIC		VARGRAPHIC
RTRIM	SYSFUN	뒤 공백을 제거한 채 인수 문자를 리턴합니다.	
	VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M)		CLOB(1M)
SECOND	SYSIBM	값의 초(시간 단위) 부분을 리턴합니다.	
	VARCHAR		INTEGER
	TIME		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
SIGN	SYSFUN	인수의 부호 표시기를 리턴합니다. 인수가 0미만일 경우, -1이 리턴됩니다. 인수가 0과 같으면 0이 리턴됩니다. 인수가 0보다 크면, 1이 리턴됩니다.	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
SIN	SYSFUN	인수의 사인을 리턴하며, 여기서 인수는 라디안으로 표시되는 각도입니다.	
	DOUBLE		DOUBLE
SMALLINT	SYSIBM	숫자의 작은(small) 정수 표현을 리턴합니다.	
	숫자 유형		SMALLINT
	VARCHAR		SMALLINT
SOUNDEX	SYSFUN	인수에서 단어들의 사운드를 나타내는 4자 코드를 리턴합니다. 그 결과는 다른 문자열의 사운드와 비교하는 데 사용됩니다. DIFFERENCE도 참조합니다.	
	VARCHAR(4000)		CHAR(4)
SPACE	SYSFUN	*인수1* 공백들로 구성된 문자열을 리턴합니다.	
	INTEGER		VARCHAR(4000)
SQLCACHE_SNAPSHOT	SYSFUN	db2 동적 SQL문 캐쉬의 스냅샷의 테이블을 리턴합니다.	
	429 페이지의 『SQLCACHE_SNAPSHOT』에서 참조하십시오.		

표 15. 지원되는 함수 (계속)

함수 이름	*스키마*	설명	
	입력 매개변수		**리턴**
SQRT	SYSFUN	인수의 제곱근을 리턴합니다.	
	DOUBLE		DOUBLE
STDDEV	SYSIBM	숫자 집합의 표준 편차를 리턴합니다(컬럼 함수).	
	DOUBLE		DOUBLE
SUBSTR	SYSIBM	*인수3* 문자에 대해 *인수2*에서 시작하는 문자열 *인수1*의 부속 문자열을 리턴합니다. *인수3*을 지정하지 않은 경우, 문자열의 나머지가 가정됩니다.	
	문자열 유형, INTEGER		*문자열 유형*
	문자열 유형, INTEGER, INTEGER		*문자열 유형*
SUM	SYSIBM	숫자 집합의 합계를 리턴합니다(컬럼 함수).	
	숫자 유형 [4]		*최대 숫자 유형*[1]
TABLE_NAME	SYSIBM	*인수1*에 제공된 오브젝트 이름에 따라 테이블이나 뷰의 규정화되지 않은 이름을 리턴하고, *인수2*에 제공된 선택적 *스키마* 이름을 리턴합니다. 이것은 별명을 해석하는 데 사용됩니다.	
	VARCHAR		VARCHAR(128)
	VARCHAR, VARCHAR		VARCHAR(128)
TABLE_SCHEMA	SYSIBM	*인수1*의 오브젝트 이름에 의해 제공되는 두 부분으로 된 테이블이나 뷰(view)의 이름 중 *스키마* 이름 부분을 리턴하고, *인수2*에 있는 옵션 *스키마* 이름을 리턴합니다. 이것은 별명을 해석하는 데 사용됩니다.	
	VARCHAR		VARCHAR(128)
	VARCHAR, VARCHAR		VARCHAR(128)
TAN	SYSFUN	인수의 탄젠트를 리턴하며, 여기서 인수는 라디안으로 표시된 각도입니다.	
	DOUBLE		DOUBLE
TIME	SYSIBM	값으로부터 시간을 리턴합니다.	
	TIME		TIME
	TIMESTAMP		TIME
	VARCHAR		TIME
TIMESTAMP	SYSIBM	하나의 값이나 한 쌍의 값으로부터 시간소인을 리턴합니다.	
	TIMESTAMP		TIMESTAMP
	VARCHAR		TIMESTAMP
	VARCHAR, VARCHAR		TIMESTAMP
	VARCHAR, TIME		TIMESTAMP
	DATE, VARCHAR		TIMESTAMP
	DATE, TIME		TIMESTAMP

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
TIMESTAMP_ISO	SYSFUN	날짜, 시간 또는 시간소인 인수에 기초하여 시간소인 값을 리턴합니다. 인수가 날짜이면, 모든 시간 요소에 대해 0을 삽입합니다. 인수가 시간이면, 날짜 요소에 대해 CURRENT DATE의 값을 삽입하고 분수 시간 요소에 대해 0을 삽입합니다.	
	DATE		TIMESTAMP
	TIME		TIMESTAMP
	TIMESTAMP		TIMESTAMP
	VARCHAR(26)		TIMESTAMP
TIMESTAMPDIFF	SYSFUN	두 시간소인의 차이에 따라 *인수1* 유형의 측정된 간격 수를 리턴합니다. 두 번째 인수는 두 개의 시간소인 유형을 빼서 결과를 CHAR로 변환한 결과입니다. 간격(*인수1*)의 유효값은 다음과 같습니다. **1** 초의 분 **2** 초 **4** 분 **8** 시간 **16** 일 **32** 주 **64** 개월 **128** 분기 **256** 년	
	INTEGER, CHAR(22)		INTEGER
TRANSLATE	SYSIBM	하나 이상의 문자가 다른 문자로 변환된 문자열을 리턴합니다.	
	CHAR		CHAR
	VARCHAR		VARCHAR
	CHAR, VARCHAR, VARCHAR		CHAR
	VARCHAR, VARCHAR, VARCHAR		VARCHAR
	CHAR, VARCHAR, VARCHAR, VARCHAR		CHAR
	VARCHAR, VARCHAR, VARCHAR, VARCHAR		VARCHAR
	GRAPHIC, VARGRAPHIC, VARGRAPHIC		GRAPHIC
	VARGRAPHIC, VARGRAPHIC, VARGRAPHIC		VARGRAPHIC
	GRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC		GRAPHIC
	VARGRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC		VARGRAPHIC

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		**리턴**
TRUNC 또는 TRUNCATE	SYSFUN	소수점의 **오른쪽** *인수2* 자리로 절단된 *인수1*을 리턴합니다. *인수2*가 음수이면, *인수1*은 소수점 **왼쪽**으로 *인수2*자리의 절대값으로 절단됩니다.	
	INTEGER, INTEGER		INTEGER
	BIGINT, INTEGER		BIGINT
	DOUBLE, INTEGER		DOUBLE
TYPE_ID [3]	SYSIBM	인수의 동적 데이터 유형의 내부 데이터 유형 식별자를 리턴합니다. 이 함수의 결과는 데이터베이스를 통해 이동가능(portable)합니다.	
	구조화 유형		INTEGER
TYPE_NAME [3]	SYSIBM	인수의 동적 데이터 유형의 규정되지 않은 이름을 리턴합니다.	
	구조화 유형		VARCHAR(18)
TYPE_SCHEMA [3]	SYSIBM	인수의 동적 유형의 스키마 이름을 리턴합니다.	
	구조화 유형		VARCHAR(128)
UCASE 또는 UPPER	SYSIBM	모든 문자가 대문자로 변환된 문자열을 리턴합니다.	
	CHAR		CHAR
	VARCHAR		VARCHAR
UCASE	SYSFUN	모든 문자가 대문자로 변환된 문자열을 리턴합니다.	
	VARCHAR		VARCHAR
VALUE [3]	SYSIBM	COALESCE와 동일.	
VARCHAR	SYSIBM	첫번째 인수의 VARCHAR 표현을 리턴합니다. 두 번째 인수가 존재하면, 결과의 길이를 지정합니다.	
	문자 유형		VARCHAR
	문자 유형, INTEGER		VARCHAR
	날짜 시간 유형		VARCHAR
VARGRAPHIC	SYSIBM	첫번째 인수의 VARGRAPHIC 표현을 리턴합니다. 두 번째 인수가 존재하면, 결과의 길이를 지정합니다.	
	그래픽 유형		VARGRAPHIC
	그래픽 유형, INTEGER		VARGRAPHIC
	VARCHAR		VARGRAPHIC
VARIANCE 또는 VAR	SYSIBM	숫자 집합의 분산도를 리턴합니다(컬럼 함수).	
	DOUBLE		DOUBLE
WEEK	SYSFUN	인수에 있는 연도의 주를 1-54 범위내의 정수 값으로 리턴합니다.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명
	입력 매개변수	**리턴**
WEEK_ISO	SYSFUN	인수에 있는 연도의 주를 1-53 범위내의 정수 값으로 리턴합니다. 일주일의 첫번째 요일은 월요일입니다. 주 1은 목요일을 포함할 연도의 첫번째 주입니다.
	VARCHAR(26)	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
YEAR	SYSIBM	값의 연 부분을 리턴합니다.
	VARCHAR	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
“+”	SYSIBM	두 개의 숫자 피연산자를 리턴합니다.
	숫자 유형, 숫자 유형	*최대 숫자 유형*
“+”	SYSIBM	단일(unary) 더하기 연산자.
	숫자 유형	*숫자 유형*
“+”	SYSIBM	날짜시간 더하기 연산자.
	DATE, DECIMAL(8,0)	DATE
	TIME, DECIMAL(6,0)	TIME
	TIMESTAMP, DECIMAL(20,6)	TIMESTAMP
	DECIMAL(8,0), DATE	DATE
	DECIMAL(6,0), TIME	TIME
	DECIMAL(20,6), TIMESTAMP	TIMESTAMP
	날짜 유형, DOUBLE, *레이블된 기간 코드*	*날짜 시간 유형*
“-”	SYSIBM	두 개의 숫자 피연산자를 뺍니다.
	숫자 유형, 숫자 유형	*최대 숫자 유형*
“-”	SYSIBM	단일(unary) 빼기 연산자.
	숫자 유형	*숫자 유형* [1]

표 15. 지원되는 함수 (계속)

함수 이름	스키마	설명	
	입력 매개변수		리턴
"-"	SYSIBM	날짜시간 빼기 연산자.	
	DATE, DATE		DECIMAL(8,0)
	TIME, TIME		DECIMAL(6,0)
	TIMESTAMP, TIMESTAMP		DECIMAL(20,6)
	DATE, VARCHAR		DECIMAL(8,0)
	TIME, VARCHAR		DECIMAL(6,0)
	TIMESTAMP, VARCHAR		DECIMAL(20,6)
	VARCHAR, DATE		DECIMAL(8,0)
	VARCHAR, TIME		DECIMAL(6,0)
	VARCHAR, TIMESTAMP		DECIMAL(20,6)
	DATE, DECIMAL(8,0)		DATE
	TIME, DECIMAL(6,0)		TIME
	TIMESTAMP, DECIMAL(20,6)		TIMESTAMP
	날짜 유형, DOUBLE, *레이블된 기간 코드*		*날짜 시간 유형*
"*"	SYSIBM	두 개의 숫자 피연산자를 곱합니다.	
	숫자 유형, *숫자 유형*		*최대 숫자 유형*
"/"	SYSIBM	두 개의 숫자 피연산자를 나눕니다.	
	숫자 유형, *숫자 유형*		*최대 숫자 유형*
"‖"	SYSIBM	CONCAT와 동일함.	

주

- 길이에 의해 규정화되지 않은 문자열 데이터 유형에 대한 참조는 해당 데이터 유형에 대해 최대 길이를 지원하는 것으로 간주해야 합니다.
- 정밀도와 스케일이 없는 DECIMAL 데이터 유형에 대한 참조는, 지원하는 정밀도와 스케일을 허용하는 것으로 간주해야 합니다.

테이블에 대한 키

모든 내장 유형 구별 유형이 아닌 모든 데이터 유형.

모든 유형 데이터베이스에 대해 정의된 모든 유형.

구조화 유형 데이터베이스에 대해 정의된 사용자 정의 구조화 유형.

비교가능 유형 109 페이지의 『지정 및 비교』에서 정의한 대로 다른 인수와 비교할 수 있는 모든 유형.

단일체계 호환가능 유형

125 페이지의 『결과 데이터 유형 규칙』에 정의된 것 처럼 다른 인수와 호환성있는 모든 유형.

문자 유형 문자열 유형: CHAR, VARCHAR, LONG VARCHAR, CLOB.

호환가능 문자열 유형

다른 인수와 같은 그룹에서 오는 문자열(예를 들어, 인수가 *문자 유형*이면 다른 것도 *문자 유형*이어야 함).

날짜 시간 유형 날짜 시간 유형: DATE, TIME, TIMESTAMP.

그래픽 유형 2바이트 문자열 유형 중 하나: GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB.

레이블된 기간 코드 이 유형은 SMALLINT입니다. 함수가 + 또는 - 연산자의 삽입 양식을 사용하여 호출된 경우, 191 페이지의 『레이블된 지속 기간』에 정의된 대로 레이블이 붙은 기간을 사용할 수 있습니다. 이름으로 +나 - 연산자를 사용하지 않는 소스 함수의 경우, 함수를 호출할 때 레이블된 기간 코드 인수에 대해 다음 값을 사용해야 합니다.

1 YEAR 또는 YEARS

2 MONTH 또는 MONTHS

3 DAY 또는 DAYS

4 HOUR 또는 HOURS

5 MINUTE 또는 MINUTES

6 SECOND 또는 SECONDS

7 MICROSECOND 또는 MICROSECONDS

LOB 유형 대형 오브젝트(LOB) 유형 중 하나: BLOB, CLOB, DBCLOB.

최대 숫자 유형 최대값이 가장 오른쪽의 *숫자 유형*으로 정의되는 인수의 최대 숫자 유형.

최대 문자열 유형 최대값이 가장 오른쪽의 *문자 유형*이나 *그래픽 유형*으로 정의되는 인수의 최대 문자열 유형. 인수가 BLOB이면, *최대 문자열 유형*은 BLOB입니다.

숫자 유형 숫자 유형으로는 SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE이 있습니다.

문자열 유형 *문자 유형*, *그래픽 유형* 또는 BLOB으로부터의 임의의 유형.

테이블 각주

[1] 입력 매개변수가 SMALLINT일 때, 결과 유형은 INTEGER입니다. 입력 매개변수가 REAL일 때 결과 유형은 DOUBLE입니다.

[2] 허용되는 키워드는 ISO, USA, EUR, JIS 및 LOCAL입니다. 연속 함수 이름은 소스 함수로 지원되지 않습니다.

[3] 이 함수는 소스 함수로 사용될 수 없습니다.

[4] 키워드 ALL 또는 DISTINCT는 첫번째 매개변수 전에 사용될 수 있습니다. DISTINCT를 지정하면, 사용자가 정의하는 구조화 유형의 사용, long 문자열 유형 또는 DATALINK 유형은 지원되지 않습니다.

[5] 사용자가 정의하는 구조화 유형의 사용, long 문자열 유형 또는 DATALINK 유형은 지원되지 않습니다.

[6] RAISE_ERROR에 의해 리턴되는 유형은 사용 문맥에 따라 다릅니다. 특정 유형으로 변환하지 않을 경우, RAISE_ERROR는 CAST 표현식 내에서의 호출에 적합한 유형을 리턴합니다.

컬럼 함수

컬럼 함수의 인수는 표현식으로부터 파생되는 일련의 값 집합입니다. 표현식에 컬럼은 포함될 수 있으나, *스칼라 fullselect* 또는 다른 컬럼 함수는 포함될 수 없습니다(SQLSTATE 42607). 집합의 범위는 433 페이지의 『제5장 조회』에 설명된 대로 그룹이거나 중간 결과 테이블입니다.

GROUP BY절이 조회에 지정되지 않고 FROM, WHERE, GROUP BY 및 HAVING절의 중간 결과가 비어 있을 경우, 컬럼 함수가 적용되지 않으며 조회 결과는 공백이며, SQLCODE는 +100으로 설정되고 SQLSTATE는 '02000'으로 설정됩니다.

GROUP BY절이 조회에서 지정되지 않고 FROM, WHERE 및 HAVING절의 중간 결과가 비어 있는 경우, 컬럼 함수는 빈 집합에 대해 적용됩니다.

예를 들어, 다음 SELECT문의 결과는 부서 D01에 있는 사원들에 대한 JOBCODE의 고유한 값의 수입니다.

```
SELECT COUNT(DISTINCT JOBCODE)
  FROM CORPDATA.EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

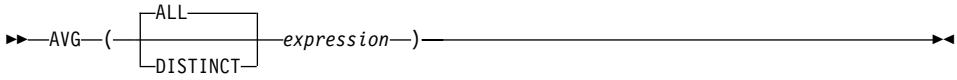
DISTINCT 키워드는 함수의 인수로 간주되지 않고, 함수가 적용되기 전에 수행되는 조작의 지정으로 간주됩니다. DISTINCT가 지정되면, 중복되는 값은 제거됩니다. ALL이 암시적이나 명시적으로 지정되면, 중복되는 값은 제거되지 않습니다.

표현식은 컬럼 함수에서 사용할 수 있습니다. 예를 들면, 다음과 같습니다.

```
SELECT MAX(BONUS + 1000)
  INTO :TOP_SALESREP_BONUS
  FROM EMPLOYEE
  WHERE COMM > 5000
```

다음에 오는 컬럼 함수는 SYSIBM 스키마에 있고 스키마 이름으로 규정화될 수 있습니다(예, SYSIBM.COUNT(*)).

AVG



스키마는 SYSIBM입니다.

AVG 함수는 숫자 집합의 평균을 리턴합니다.

인수값은 숫자여야 하고 그 합은 결과의 데이터 유형 범위 내에 있어야 합니다. 결과는 널(NULL)이 될 수 있습니다.

결과의 데이터 유형은 다음의 경우를 제외하고 인수 값의 데이터 유형과 같습니다.

- 인수 값이 작은 정수일 경우 결과는 큰 정수입니다.
- 인수 값이 단일 정밀도의 부동 소수점일 경우 결과는 배정밀도의 부동 소수점입니다.

인수 값의 데이터 유형이 정밀도가 *p*이고 스케일이 *s*인 십진수인 경우, 결과의 정밀도는 31이고, 스케일은 31-*p*+*s*입니다.

함수는 널(NULL) 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다. DISTINCT가 지정되면, 중복되는 값은 제거됩니다.

함수가 공집합에 적용되면, 결과는 널(NULL) 값입니다. 그렇지 않으면, 그 결과는 집합의 평균 값입니다.

결과의 유형이 정수이면, 평균의 소수 부분은 유실됩니다.

예:

- PROJECT 테이블을 사용하여, 호스트 변수 AVERAGE(소수(5,2))를 부서(DEPTNO) 'D11'에서의 프로젝트에 대한 평균 스태핑 레벨(PRSTAFF)로 설정합니다.

```
SELECT AVG(PRSTAFF)
  INTO :AVERAGE
                FROM PROJECT
  WHERE DEPTNO = 'D11'
```

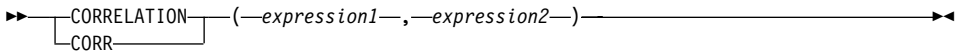
샘플 테이블을 사용할 경우 AVERAGE가 4.25(즉 17/4)로 설정됩니다.

- PROJECT 테이블을 사용하여, 호스트 변수 ANY_CALC(십진수(5,2))를 부서 (DEPTNO) 'D11'에서의 프로젝트에 대한 각 고유 스태핑 레벨 값(PRSTAFF) 평균으로 설정합니다.

```
SELECT AVG(DISTINCT PRSTAFF)
  INTO :ANY_CALC
              FROM PROJECT
  WHERE DEPTNO = 'D11'
```

샘플 테이블을 사용할 경우 ANY_CALC가 4.66(즉, 14/3)으로 설정됩니다.

CORRELATION



스키마는 SYSIBM입니다.

CORRELATION 함수는 수 쌍 집합의 상관의 계수를 리턴합니다.

인수값은 숫자여야 합니다.

결과의 데이터 유형은 배정밀도 부동 소수점 수입니다. 결과는 널(NULL)이 될 수 있습니다. 널(NULL)이 아닐 경우, 결과는 0과 1 사이입니다.

함수는 *expression1*이나 e*xpression2*가 널(NULL)인 모든 쌍을 제거하여 인수 값으로부터 파생된 (*expression1*, *expression2*) 쌍의 집합에 적용됩니다.

함수가 공집합에 적용되거나, STDDEV(*expression1*) 또는 STDDEV(*expression2*)가 0이면, 결과는 널(NULL) 값입니다. 그렇지 않으면, 결과는 집합에 있는 값 쌍에 대한 상관 계수입니다. 결과는 다음에 표현식에 해당합니다.

```
COVARIANCE(expression1,expression2)/(STDDEV(expression1)*STDDEV(expression2
```

값이 더해지는 순서는 정의되어 있지 않지만, 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

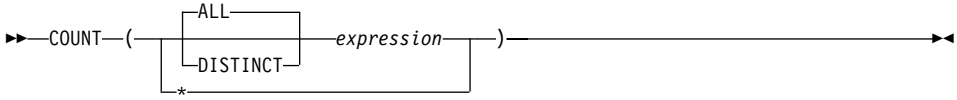
예:

- EMPLOYEE 테이블을 사용하여, 호스트 변수 CORRLN(배정밀도의 부동 소수점 수)를 부서(WORKDEPT) 'A00'에 있는 직원에 대한 급여와 보너스 사이의 상관으로 설정하십시오.

  ```
  SELECT CORRELATION (SALARY, BONUS)
    INTO :CORRLN
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  샘플 테이블 사용시 CORRLN는 대략 9.99853953399538E-001로 설정됩니다.

COUNT



스키마는 SYSIBM입니다.

COUNT 함수는 행이나 값 집합에 행이나 값의 개수를 리턴합니다.

DISTINCT를 사용할 경우, *표현식*의 결과 데이터 유형은 문자 컬럼의 경우 255, 그래픽 컬럼의 경우 127 길이를 초과할 수 없습니다. *표현식*의 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, 이 유형들 중 하나에 대한 구별 유형, 또는 구조화 유형이 될 수 없습니다(SQLSTATE 42907).

함수의 결과는 큰 정수입니다. 결과는 널(NULL)이 될 수 없습니다.

COUNT(*)의 인수는 행의 집합입니다. 결과는 집합 내의 행 수입니다. NULL 값만을 포함하는 행이 개수에 포함됩니다.

COUNT(DISTINCT *표현식*)의 인수는 값 집합입니다. 함수는 널(NULL) 값과 중복 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다. 결과는 집합 내의 널(NULL)이 아닌 서로 다른 값들의 개수입니다.

COUNT(*표현식*)이나 COUNT(ALL *표현식*)의 인수는 값 집합입니다. 함수는 널(NULL) 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다. 결과는 중복값을 포함하여, 집합 내의 널(NULL)이 아닌 값들의 개수입니다.

예:

- EMPLOYEE 테이블을 사용하여, 호스트 변수 FEMALE(int)를 SEX 컬럼의 값이 'F'인 행 수로 설정합니다.

```
SELECT COUNT(*)
  INTO :FEMALE
                          FROM EMPLOYEE
  WHERE SEX = 'F'
```

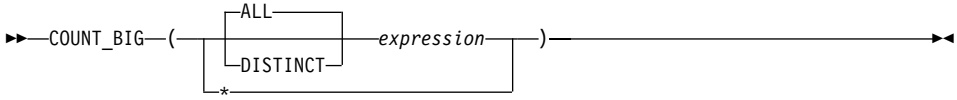
샘플 테이블을 사용할 경우 FEMALE가 13으로 설정됩니다.

- EMPLOYEE 테이블을 사용하여, 호스트 변수 FEMALE_IN_DEPT(int)를, 구성원으로 최소한 하나의 F를 갖는 부서 수(WORKDEPT)로 설정합니다.

```
SELECT COUNT(DISTINCT WORKDEPT)
  INTO :FEMALE_IN_DEPT
                              FROM EMPLOYEE
  WHERE SEX = 'F'
```

샘플 테이블을 사용할 경우 FEMALE_IN_DEPT가 5로 설정됩니다(A00, C01, D11, D21, E11 부서에 최소한 한 명의 여사원이 있습니다.).

COUNT_BIG



스키마는 SYSIBM입니다.

COUNT_BIG 함수는 행이나 값의 수를 행이나 값 집합으로 리턴합니다. 이것은 결과가 정수의 최대 값보다 클 수도 있다는 점을 제외하면 COUNT와 유사합니다.

DISTINCT를 사용할 경우, *표현식*의 결과 데이터 유형은 문자 컬럼의 경우 255, 그래픽 컬럼의 경우 127 길이를 초과할 수 없습니다. *표현식*의 결과 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, 이 유형들 중 하나에 대한 구별 유형, 또는 구조화 유형이 될 수 없습니다(SQLSTATE 42907).

함수의 결과는 정밀도 31, 스케일 0인 십진수입니다. 결과는 널(NULL)이 될 수 없습니다.

COUNT_BIG(*)의 인수는 행 집합입니다. 결과는 집합 내의 행 수입니다. NULL 값만을 포함하는 행이 개수에 포함됩니다.

COUNT_BIG(DISTINCT *표현식*)의 인수는 값 집합입니다. 함수는 널(NULL) 값과 중복 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다. 결과는 집합 내의 널(NULL)이 아닌 서로 다른 값들의 개수입니다.

COUNT_BIG(*표현식*) 또는 COUNT_BIG(ALL *표현식*)의 인수는 값 집합입니다. 함수는 널(NULL) 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다. 결과는 중복값을 포함하여, 집합 내의 널(NULL)이 아닌 값들의 개수입니다.

예:

- COUNT의 쓰임에 대해서는 COUNT의 예와 대안인 COUNT_BIG을 참조하십시오. 데이터 유형을 제외하면 결과들은 동일합니다.

- 일부 응용프로그램에서는 COUNT를 사용해야 하지만 최대 정수보다 더 큰 값을 지원해야 할 수 있습니다. 이런 경우에는 전래 사용자 정의 함수를 사용하고 SQL 경로를 설정하면 됩니다. 다음의 일련의 명령문은 COUNT_BIG에 기초하여 COUNT(*)를 지원하기 위한 전래 함수의 작성 방식과 정밀도 15의 십진값을 리턴하는 방법을 보여줍니다. SQL 경로는 COUNT_BIG에 기초한 전래 함수가 제시된 조회에서와 같이 후속 명령문으로 사용되도록 설정됩니다.

```
CREATE FUNCTION RICK.COUNT() RETURNS DECIMAL(15,0)
       SOURCE SYSIBM.COUNT_BIG();
SET CURRENT FUNCTION PATH RICK, SYSTEM PATH;
SELECT COUNT(*) FROM EMPLOYEE;
```

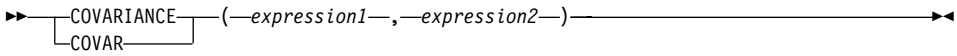
  COUNT(*)를 지원하는 매개변수가 없는 경우 전래 함수의 정의 방식에 주의하십시오. 이는 함수 COUNT를 명명하였지만 이 명령을 사용할 때 스키마 이름을 사용하여 함수를 규정하지 않은 경우에만 적용됩니다. 이름을 COUNT로 하지 않은 경우 COUNT(*)와 동일한 효과를 얻으려면, 매개변수없이 함수를 호출하십시오. 따라서, RICK.COUNT가 RICK.MYCOUNT로 정의된 경우, 다음과 같이 조회를 작성해야 합니다.

```
SELECT MYCOUNT() FROM EMPLOYEE;
```

  특정 컬럼에 대해 count 함수가 사용되는 경우, 전래 함수가 컬럼의 유형을 지정해야 합니다. 전래 함수를 작성한 다음 명령문은 임의의 CHAR 컬럼을 인수로 사용하고 COUNT_BIG을 계수를 수행하는 데 사용합니다.

```
CREATE FUNCTION RICK.COUNT(CHAR()) RETURNS DOUBLE
       SOURCE SYSIBM.COUNT_BIG(CHAR());
SELECT COUNT(DISTINCT WORKDEPT) FROM EMPLOYEE;
```

COVARIANCE



스키마는 SYSIBM입니다.

COVARIANCE 함수는 수 쌍 집합의 공분산을 리턴합니다.

인수값은 숫자여야 합니다.

결과의 데이터 유형은 배정밀도 부동 소수점 수입니다. 결과는 널(NULL)이 될 수 있습니다.

함수는 *expression1*이나 e*xpression2*가 널(NULL)인 모든 쌍을 제거하여 인수 값으로부터 파생된 (*expression1*, *expression2*) 쌍의 집합에 적용됩니다.

함수가 공집합에 적용되면, 결과는 널(NULL) 값입니다. 그렇지 않으면, 결과는 집합에 있는 값 쌍의 공분산입니다. 결과는 다음에 해당합니다.

1. avgexp1이 AVG(*expression1*)의 결과이고 avgexp2가 AVG(*expression2*)의 결과라고 합시다.
2. COVARIANCE(*expression1*, *expression2*)의 결과는 AVG((*expression1* - avgexp1) * (*expression2* - avgexp2)입니다.

값이 더해지는 순서는 정의되어 있지 않지만, 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

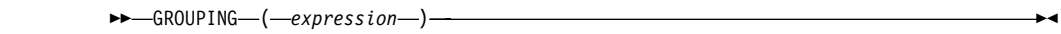
예:

- EMPLOYEE 테이블을 사용하여, 호스트 변수 COVARNCE(배정밀도의 부동 소수점 수)를 부서(WORKDEPT) 'A00'에 있는 직원에 대한 급여와 보너스 사이의 공분산으로 설정하십시오.

```
SELECT COVARIANCE(SALARY, BONUS)
  INTO :COVARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00'
```

샘플 테이블 사용시 COVARNCE는 대략 1.68888888888889E+006으로 설정됩니다.

GROUPING



스키마는 SYSIBM입니다.

그룹화 집합(grouping-sets)과 수퍼 그룹(super-groups)을 (자세한 사항은 449 페이지의 『group-by절』 참조) 결합하여 사용할 때, GROUPING 함수는 GROUP BY 대답 집합(answer set)에서 리턴되는 행이 *expression*으로 나타나는 컬럼을 제거하는, 그룹화 집합에 의해 생성 가능 여부를 지시하는 값을 리턴합니다.

인수는 어느 유형이나 가능하지만, GROUP BY절의 항목이어야 합니다.

함수의 결과는 작은 정수입니다. 이는 다음 값 중 하나로 설정됩니다.

1 리턴된 행에서 *표현식*의 값은 널(NULL) 값이고, 행은 수퍼 그룹(super-group)이 생성합니다. 이렇게 생성된 행은 GROUP BY 표현식에 대한 부분 합계 값을 제공하는 데 사용할 수 있습니다.

0 위의 값이외의 값입니다.

예:

조회:

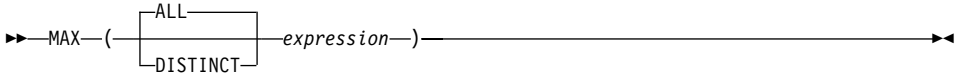
```
SELECT SALES_DATE,
       SALES_PERSON,
       SUM(SALES) AS UNITS_SOLD,
       GROUPING(SALES_DATE) AS DATE_GROUP,
       GROUPING(SALES_PERSON) AS  SALES_GROUP
FROM SALES
  GROUP BY CUBE(SALES_DATE, SALES_PERSON)
  ORDER BY SALES_DATE, SALES_PERSON
```

결과:

```
SALES_DATE SALES_PERSON    UNITS_SOLD  DATE_GROUP  SALES_GROUP
---------- --------------- ----------- ----------- -----------
12/31/1995 GOUNOT                    1           0           0
12/31/1995 LEE                       6           0           0
12/31/1995 LUCCHESSI                 1           0           0
12/31/1995 -                         8           0           1
03/29/1996 GOUNOT                   11           0           0
03/29/1996 LEE                      12           0           0
03/29/1996 LUCCHESSI                 4           0           0
03/29/1996 -                        27           0           1
03/30/1996 GOUNOT                   21           0           0
03/30/1996 LEE                      21           0           0
03/30/1996 LUCCHESSI                 4           0           0
03/30/1996 -                        46           0           1
03/31/1996 GOUNOT                    3           0           0
03/31/1996 LEE                      27           0           0
03/31/1996 LUCCHESSI                 1           0           0
03/31/1996 -                        31           0           1
04/01/1996 GOUNOT                   14           0           0
04/01/1996 LEE                      25           0           0
04/01/1996 LUCCHESSI                 4           0           0
04/01/1996 -                        43           0           1
-          GOUNOT                   50           1           0
-          LEE                      91           1           0
-          LUCCHESSI                14           1           0
-          -                       155           1           1
```

응용프로그램은 DATE_GROUP의 값이 0이고 SALES_GROUP의 값이 1이면 SALES_DATE 소계(sub-total) 행을 인식할 수 있습니다. DATE_GROUP의 값이 1이고 SALES_GROUP의 값이 0이라는 사실에 의해 SALES_PERSON 소계(sub-total) 행을 인식할 수 있습니다. 총계 행은 DATE_GROUP과 SALES_GROUP의 값 1에 의해 인식될 수 있습니다.

MAX



스키마는 SYSIBM입니다.

MAX 함수는 값 집합에서 최대값을 리턴합니다.

인수 값은 long 문자열 또는 DATALINK를 제외한 모든 내장 유형이 될 수 있습니다.

DISTINCT를 사용할 경우, *표현식*의 결과 데이터 유형은 문자 컬럼의 경우 255, 그래픽 컬럼의 경우 127 길이를 초과할 수 없습니다. *표현식*의 결과 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, 이 유형들 중 하나에 대한 구별 유형, 또는 구조화 유형이 될 수 없습니다(SQLSTATE 42907).

결과의 데이터 유형, 길이 및 코드 페이지는 인수 값의 데이터 유형, 길이 및 코드 페이지와 같습니다. 결과는 도출된 값으로 간주되고 널(NULL)이 될 수 있습니다.

함수는 널(NULL) 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다.

함수가 공집합에 적용되면, 결과는 널(NULL) 값입니다. 그렇지 않으면, 결과는 집합 내에서 최고값이 됩니다.

DISTINCT를 지정해도 그 결과에는 아무런 영향도 미치지 않으므로, 사용하지 않는 것이 바람직합니다. 이것은 다른 관계형 시스템과의 호환성을 위해 포함됩니다.

예:

- EMPLOYEE 테이블을 사용하여, 호스트 변수 MAX_SALARY(decimal(7,2))를 최대 월 급여(SALARY/12) 값으로 설정합니다.

```
SELECT MAX(SALARY) / 12
  INTO :MAX_SALARY
                        FROM EMPLOYEE
```

  샘플 테이블을 사용할 경우 MAX_SALARY가 4395.83으로 설정됩니다.

- PROJECT 테이블을 사용하여, 호스트 변수 LAST_PROJ(char(24))를 조합 순서에서 마지막으로 제공되는 프로젝트 이름(PROJNAME)으로 설정합니다.

```
SELECT MAX(PROJNAME)
  INTO :LAST_PROJ
               FROM PROJECT
```

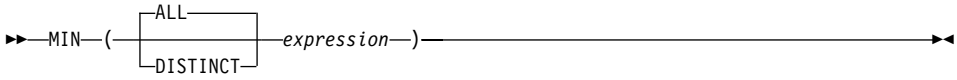
  샘플 테이블을 사용할 경우 LAST_PROJ가 'WELD LINE PLANNING'으로 설정됩니다.

- 이전 예와 마찬가지로, 호스트 변수 LAST_PROJ(char(40))를, 프로젝트 이름이 호스트 변수 PROJSUPP와 연결될 경우 조합 순서에서 마지막으로 제공되는 프로젝트 이름(PROJNAME)으로 설정합니다. PROJSUPP는 '_Support'이고, 데이터 유형은 char(8)입니다.

```
SELECT MAX(PROJNAME CONCAT PROJSUPP)
  INTO :LAST_PROJ
               FROM PROJECT
```

  샘플 테이블을 사용할 경우 LAST_PROJ가 'WELD LINE PLANNING_SUPPORT'로 설정됩니다.

MIN



스키마는 SYSIBM입니다.

MIN 함수는 값 집합에서 최소값을 리턴합니다.

인수 값은 long 문자열 또는 DATALINK를 제외한 모든 내장 유형이 될 수 있습니다.

DISTINCT를 사용할 경우, *표현식*의 결과 데이터 유형은 문자 컬럼의 경우 255, 그래픽 컬럼의 경우 127 길이를 초과할 수 없습니다. *표현식*의 결과 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, 이 유형들 중 하나에 대한 구별 유형, 또는 구조화 유형이 될 수 없습니다(SQLSTATE 42907).

결과의 데이터 유형, 길이 및 코드 페이지는 인수 값의 데이터 유형, 길이 및 코드 페이지와 같습니다. 결과는 도출된 값으로 간주되고 널(NULL)이 될 수 있습니다.

함수는 널(NULL) 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다.

이 함수가 공집합에 적용되면, 결과는 널(NULL) 값입니다. 그렇지 않으면, 결과는 집합에서 최소값이 됩니다.

DISTINCT를 지정해도 그 결과에는 아무런 영향도 미치지 않으므로, 사용하지 않는 것이 바람직합니다. 이것은 다른 관계형 시스템과의 호환성을 위해 포함됩니다.

예:

- EMPLOYEE 테이블을 사용하여, 호스트 변수 COMM_SPREAD (decimal(7,2))를 부서 (WORKDEPT) 'D11' 구성원에 대한 최대 및 최소 수당(COMM) 사이의 차이로 설정합니다.

```
SELECT MAX(COMM) - MIN(COMM)
  INTO :COMM_SPREAD
                          FROM EMPLOYEE
  WHERE WORKDEPT = 'D11'
```

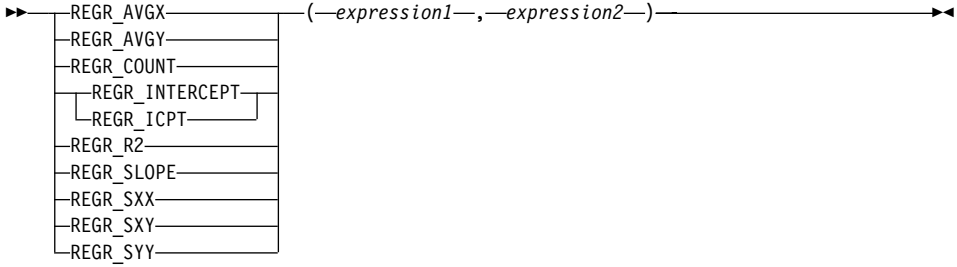
샘플 테이블을 사용할 경우 COMM_SPREAD가 1118(즉, 2580 - 1462)로 설정됩니다.

- PROJECT 테이블을 사용하여, 호스트 변수 (FIRST_FINISHED (char(10))를 처음으로 완료되도록 스케줄된 프로젝트의 예측 종료 날짜(PRENDATE)로 설정합니다.

```
SELECT MIN(PRENDATE)
  INTO :FIRST_FINISHED
               FROM PROJECT
```

샘플 테이블을 사용할 경우 FIRST_FINISHED가 '1982-09-15'로 설정됩니다.

REGRESSION 함수



스키마는 SYSIBM입니다.

역행 함수는 y = a * x + b 형태의 ordinary-least-squares 역행 라인을 일련의 수 쌍으로 맞추는 것을 지원합니다. 각 쌍의 첫번째 요소(*expression1*)는 종속 변수(즉, ″y 값″)의 값으로 해석됩니다. 각 쌍의 두 번째 요소(*expression2*)는 종속 변수(즉, ″x 값″)의 값으로 해석됩니다.

함수 REGR_COUNT는 역행 라인을 맞추는 데 사용된 널(NULL)이 아닌 수 쌍의 갯수를 리턴합니다(아래 참조).

함수 REGR_INTERCEPT (축약형은 REGR_ICPT)는 역행 라인의 y-절편을 리턴합니다(위의 등식에서 ″b″).

함수 REGR_R2는 역행에 대한 결정의 계수(″R-squared″ 또는 ″goodness-of-fit″라고도 함)를 리턴합니다.

함수 REGR_SLOPE는 행의 기울기(위의 등식에서 매개변수 ″a″)를 리턴합니다.

함수 REGR_AVGX, REGR_AVGY, REGR_SXX, REGR_SYY 및 REGR_SXY는 역행 모델의 성질 및 통계 유효성의 평가에 필요한 여러 가지 진단 통계를 계산하는 데 사용될 수 있는 분량을 리턴합니다(아래 참조).

인수값은 숫자여야 합니다.

REGR_COUNT 결과의 데이터 유형은 정수입니다. 나머지 함수의 경우, 결과의 데이터 유형은 이중 정밀도 부동 소수점입니다. 결과는 널(NULL)이 될 수 있습

니다. 모두 널(NULL)이 아닐 때에, REGR_R2의 결과는 0과 1 사이이며 REGR_SXX와 REGR_SYY 둘다의 결과는 음이 아닌 정수입니다.

각 함수는 *expression1*이나 e*xpression2*가 널(NULL)인 모든 쌍을 제거하여 인수 값으로부터 파생된 (*expression1*, *expression2*) 쌍의 집합에 적용됩니다.

집합이 비워있지 않고 VARIANCE(*expression2*)가 양수이면, REGR_COUNT가 집합에서 널(NULL)이 아닌 쌍들의 갯수를 리턴하며, 나머지 함수들은 다음과 같이 정의되는 결과를 리턴합니다.

```
REGR_SLOPE(expression1,expression2) =
 COVARIANCE(expression1,expression2)/VARIANCE(expression2)

REGR_INTERCEPT(expression1, expression2) =
  AVG(expression1) - REGR_SLOPE(expression1, expression2) * AVG(expression2)

REGR_R2(expression1, expression2) =
  POWER(CORRELATION(expression1, expression2 ), 2) if VARIANCE(expression1)>0
  REGR_R2(expression1, expression2) = 1 if VARIANCE(expression1)=0

 REGR_AVGX(expression1, expression2) =  AVG(expression2)

REGR_AVGY(expression1, expression2) = AVG(expression1)

REGR_SXX(expression1,  expression2) =
  REGR_COUNT(expression1, expression2) * VARIANCE(expression2)

 REGR_SYY(expression1, expression2) =
  REGR_COUNT(expression1, expression2) * VARIANCE(expression1)

 REGR_SXY(expression1, expression2) =
  REGR_COUNT(expression1, expression2) * COVARIANCE(expression1, expression2)
```

집합이 비워있지 않으며 VARIANCE(*expression2*)가 0이면, 역행 라인이 무한 기울기를 가지거나 미정의됩니다. 이 경우, 함수 REGR_SLOPE, REGR_INTERCEPT 및 REGR_R2는 각각 널(NULL) 값을 리턴하면, 나머지 함수들은 위에서 정의된 대로의 값들을 리턴합니다. 집합이 비워 있으면, REGR_COUNT가 0을 리턴하며 나머지 함수들은 널(NULL) 값을 리턴합니다.

값이 더해지는 순서는 정의되어 있지 않지만, 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

역행 함수는 데이터를 단일 통과하는 동안 모두 동시에 계산됩니다. 일반적으로, AVERAGE, VARIANCE, COVARIANCE 등과 같은 일반 컬럼 함수를 사용하여 해당 계산을 수행하는 것보다 역행 분석에 필요한 통계를 계산하기 위해 역행 함수를 사용하는 것이 더 효율적입니다.

선형 역행 분석에 수반된 보통의 진단 통계가 위의 함수에 의하여 계산될 수 있습니다. 예를 들면,

조정된 R2

1 - ((1 - REGR_R2) * ((REGR_COUNT - 1) / (REGR_COUNT - 2)))

표준 오류

SQRT((REGR_SYY-(POWER(REGR_SXY,2)/REGR_SXX))/ (REGR_COUNT-2))

면적의 총 합

REGR_SYY

면적의 역행 합

POWER(REGR_SXY,2) / REGR_SXX

면적의 나머지 합

(면적의 총 합)-(면적의 역행 합)

기울기에 대한 t 통계치

REGR_SLOPE * SQRT(REGR_SXX) / (표준 오류)

y-절편에 대한 t 통계치

REGR_INTERCEPT/((Standard error)*SQRT((1/REGR_COUNT)+ (POWER(REGR_AVGX,2)/REGR_SXX))

예:

- EMPLOYEE 테이블을 사용하여, 부서 (WORKDEPT) 'A00'에 있는 직원의 보너스를 직원 급여의 선형 함수로서 표현하는 ordinary-least-squares 역행 라인을 계산하십시오. 호스트 변수 SLOPE, ICPT, RSQR (이중 정밀도 부동 소수점)을 각각 역행 라인 결정의 기울기, 절편 및 계수에 설정하십시오. 또한 호스트 변수 AVGSAL 및 AVGBONUS를 각각 부서 'A00'에 있는 직원의 평

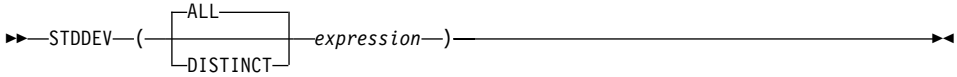
균 급여와 평균 보너스에 설정하고, 호스트 변수 CNT (정수)를 급여와 보너스 데이터 둘다가 사용 가능한 부서 'A00'에 있는 직원 번호에 설정하십시오. 나머지 역행 통계를 호스트 변수 SXX, SYY 및 SXY에 보관하십시오.

```
SELECT REGR_SLOPE(BONUS,SALARY), REGR_INTERCEPT(BONUS,SALARY),
   REGR_R2(BONUS,SALARY), REGR_COUNT(BONUS,SALARY),
   REGR_AVGX(BONUS,SALARY), REGR_AVGY(BONUS,SALARY),
   REGR_SXX(BONUS,SALARY), REGR_SYY(BONUS,SALARY),
   REGR_SXY(BONUS,SALARY)
   INTO :SLOPE, :ICPT,
        :RSQR, :CNT,
        :AVGSAL, :AVGBONUS,
        :SXX, :SYY,
        :SXY
   FROM EMPLOYEE
  WHERE WORKDEPT = 'A00'
```

샘플 테이블 사용시, 호스트 변수는 다음의 대략적인 값에 설정됩니다.

```
SLOPE:  +1.71002671916749E-002
ICPT:  +1.00871888623260E+002
RSQR:  +9.99707928128685E-001
CNT:  3
AVGSAL:  +4.28333333333333E+004
AVGBONUS:  +8.33333333333333E+002
SXX:  +2.96291666666667E+008
SYY:  +8.66666666666667E+004
SXY:  +5.06666666666667E+006
```

STDDEV



스키마는 SYSIBM입니다.

STDDEV 함수는 숫자 세트의 표준 편차를 리턴합니다.

인수값은 숫자여야 합니다.

결과의 데이터 유형은 배정밀도 부동 소수점 수입니다. 결과는 널(NULL)이 될 수 있습니다.

함수는 널(NULL) 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다. DISTINCT가 지정되면, 중복되는 값은 제거됩니다.

함수가 공집합에 적용되면, 결과는 널(NULL) 값입니다. 그렇지 않으면, 결과는 집합 내에 있는 값의 표준 편차입니다.

값이 더해지는 순서는 정의되어 있지 않지만, 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

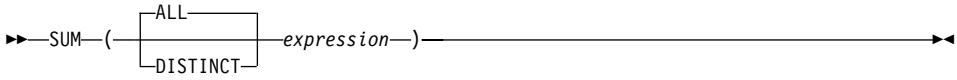
예:

- EMPLOYEE 테이블을 사용하여 호스트 변수 DEV(배정밀도의 부동 소수점 수)를 부서(WORKDEPT) 'A00'의 사원에 대한 급여의 표준 편차로 설정하십시오.

```
SELECT STDDEV(SALARY)
  INTO :DEV
                          FROM EMPLOYEE
  WHERE WORKDEPT = 'A00'
```

  샘플 테이블을 사용할 경우, DEV는 약 9938.00으로 설정됩니다.

SUM



스키마는 SYSIBM입니다.

SUM 함수는 숫자 집합의 합을 리턴합니다.

인수값은 숫자여야 하고(내장 유형 전용), 그 합은 결과의 데이터 유형 범위 내에 있어야 합니다.

결과의 데이터 유형은 다음의 경우를 제외하고 인수 값의 데이터 유형과 같습니다.

- 인수 값이 작은 정수일 경우 결과는 큰 정수입니다.
- 인수 값이 단일 정밀도의 부동 소수점일 경우 결과는 배정밀도의 부동 소수점입니다.

인수 값의 데이터 유형이 십진수이면, 결과의 정밀도가 31이고, 스케일은 인수 값의 스케일과 같습니다. 결과는 널(NULL)이 될 수 있습니다.

함수는 널(NULL) 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다. DISTINCT가 지정되면, 중복되는 값은 제거됩니다.

함수가 공집합에 적용되면, 결과는 널(NULL) 값입니다. 그렇지 않으면, 결과는 집합 내에 있는 값들의 합입니다.

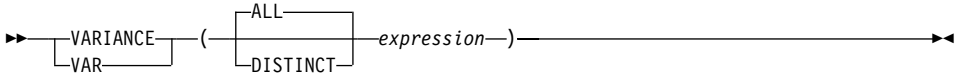
예:

- EMPLOYEE 테이블을 사용하여, 호스트 변수 JOB_BONUS (decimal(9,2))를 사원(JOB='CLERK')에게 지불된 총 보너스(BONUS)로 설정합니다.

```
SELECT SUM(BONUS)
  INTO :JOB_BONUS
                          FROM EMPLOYEE
  WHERE JOB = 'CLERK'
```

  샘플 테이블을 사용할 경우 JOB_BONUS가 2800으로 설정됩니다.

VARIANCE



스키마는 SYSIBM입니다.

VARIANCE 함수는 숫자 집합의 분산을 리턴합니다.

인수값은 숫자여야 합니다.

결과의 데이터 유형은 배정밀도 부동 소수점 수입니다. 결과는 널(NULL)이 될 수 있습니다.

함수는 널(NULL) 값이 제거된 인수 값으로부터 파생되는 값 집합에 적용됩니다. DISTINCT가 지정되면, 중복되는 값은 제거됩니다.

함수가 공집합에 적용되면, 결과는 널(NULL) 값입니다. 그렇지 않으면, 결과는 집합 내의 값의 분산입니다.

값이 더해지는 순서는 정의되어 있지 않지만, 모든 중간 결과는 결과 데이터 유형의 범위 내에 있어야 합니다.

예:

- EMPLOYEE 테이블을 사용하여 호스트 변수 VARNCE(배정밀도 부동 소수점 수)를 부서(WORKDEPT) 'A00'의 사원에 대한 급여의 분산으로 설정하십시오.

```
SELECT VARIANCE(SALARY)
  INTO :VARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00'
```

  샘플 테이블을 사용할 경우, VARNCE는 약 98763888.88로 설정됩니다.

스칼라 함수

스칼라 함수(scalar functions)는 표현식이 사용될 수 있는 곳이면 어디에서든지 사용할 수 있습니다. 그러나, 표현식과 컬럼 함수의 사용에 적용되는 제한사항은 표현식이나 컬럼 함수가 스칼라 함수 내에 사용될 때도 적용됩니다. 예를 들어, 스칼라 함수의 인수는 스칼라 함수가 사용되는 문맥에서 컬럼 함수가 허용될 경우에만 컬럼 함수가 될 수 있습니다.

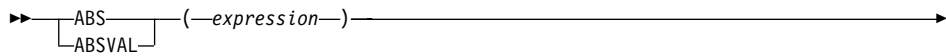
컬럼 함수 사용에 대한 제한사항은 스칼라 함수에 적용되지 않습니다. 스칼라 함수는 값의 집합이 아닌, 단일 값에 적용되기 때문입니다.

예: 다음 SELECT문의 결과에는 부서 D01에 있는 사원 수만큼의 행이 있습니다.

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BRTHDATE)
  FROM EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

다음에 오는 스칼라 함수는 스키마 이름(예: SYSIBM.CHAR(123))으로 규정화될 수 있습니다.

ABS 또는 ABSVAL



스키마는 SYSFUN입니다.

인수의 절대 값을 리턴합니다.

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 인수가 DECIMAL 또는 REAL 유형인 경우, 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 다음과 같습니다.

- 인수가 SMALLINT이면 SMALLINT입니다.
- 인수가 INTEGER이면 INTEGER입니다.
- 인수가 BIGINT이면 BIGINT입니다.
- 인수가 DOUBLE, DECIMAL 또는 REAL일 경우, DOUBLE입니다[39].

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

39. 또한, 인수가 BIGINT의 가장 작은 값, -9 223 372 036 854 775 808일 경우 DOUBLE을 리턴합니다.

ACOS

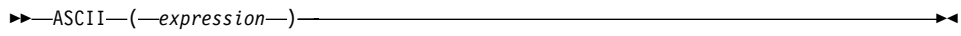
스키마는 SYSFUN입니다.

인수의 아크코사인(arccosine)을 라디안으로 표시되는 각도로 리턴합니다.

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 이는 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 &dpfpn;입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

ASCII



스키마는 SYSFUN입니다.

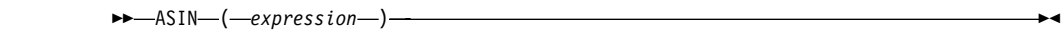
인수의 가장 왼쪽 문자의 ASCII 코드 값을 정수로 리턴합니다.

인수는 내장형 문자열 유형이면 어느 것이나 가능합니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB의 최대 길이는 1 048 576바이트입니다. 함수가 처리할 수 있도록 LONG VARCHAR가 CLOB로 변환됩니다.

함수의 결과는 반드시 INTEGER입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

ASIN



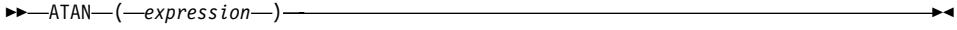
스키마는 SYSFUN입니다.

인수의 아크사인(arcsine)을 라디안으로 표시되는 각도로 리턴합니다.

인수는 내장형 숫자 유형이면 어느 것이나 가능합니다. 이는 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

ATAN



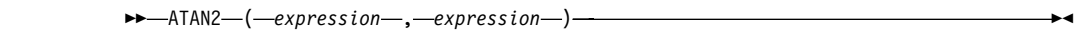
스키마는 SYSFUN입니다.

인수의 아크탄젠트를 라디안으로 표시되는 각도로 리턴합니다.

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 이는 함수가 처리할 수 있도록 &dpfpn;로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

ATAN2



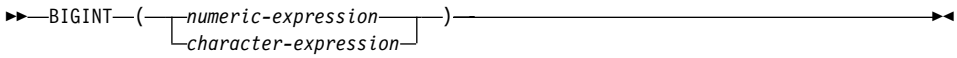
스키마는 SYSFUN입니다.

x와 y 좌표의 아크탄젠트(arctangent)를 라디안으로 표시되는 각도로 리턴합니다. x와 y 좌표는 각각 첫번째와 두 번째 인수로 지정됩니다.

첫번째와 두 번째 인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 이들 두 인수 모두 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 &dpfpn;입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

BIGINT



스키마는 SYSIBM입니다.

BIGINT 함수는 정수 상수 형식의 숫자 또는 문자에 대해 64비트 정수 표현을 리턴합니다.

숫자 표현식

임의의 내장 숫자 데이터 유형값을 리턴시키는 표현식.

인수가 *숫자 표현식*이면, 결과는 인수가 큰 정수 컬럼 또는 변수에 할당되었을 경우에 발생한 것과 같은 숫자입니다. 인수의 정수 부분이 정수 범위 내에 있지 않으면, 오류가 발생합니다. 인수의 소수 부분이 있으면, 절단됩니다.

문자 표현식

문자 상수의 최대 길이보다 잦은 길이를 갖는 문자열 값을 리턴하는 표현식입니다. 선행 공백과 뒷 공백은 제거되고 결과 문자열은 SQL 정수 상수를 형성하는 규칙에 따라야 합니다(SQLSTATE 22018). 문자열은 long 문자열이 될 수 없습니다.

인수가 *문자 표현식*이면, 결과는 해당 정수 상수가 큰 정수 컬럼 또는 변수에 할당되었을 경우에 발생한 것과 같은 수입니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예:

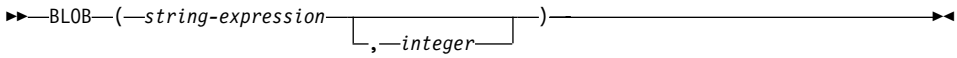
- ORDERS_HISTORY 테이블에서, 순서 번호를 합계한 후 결과를 큰 정수 값으로서 리턴합니다.

```
SELECT BIGINT (COUNT_BIG(*) )
  FROM ORDERS_HISTORY
```

- EMPLOYEE 테이블을 사용하여 적용업무를 추가로 처리하려면 큰 정수 양식에서 EMPNO 컬럼을 선택하십시오.

```
SELECT BIGINT(EMPNO) FROM EMPLOYEE
```

BLOB



스키마는 SYSIBM입니다.

BLOB 함수는 임의 유형 문자열의 BLOB 표현을 리턴합니다.

문자열 표현식

값이 문자열, 그래픽 문자열 또는 2진 문자열이 될 수 있는 *문자열 표현식*

정수

결과로 나오는 BLOB 데이터 유형의 길이 속성을 지정하는 정수 값. *정수*를 지정하지 않으면, 결과의 길이 속성은 입력이 그래픽인 위치를 제외하고는 입력 길이와 동일합니다. 이 경우, 결과의 길이 속성은 입력 길이의 두 배입니다.

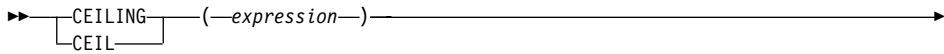
함수의 결과는 BLOB입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예:

- TOPOGRAPHIC_MAP으로 명명된 BLOB 컬럼과 MAP_NAME으로 명명된 VARCHAR 컬럼이 있는 테이블이 제공된 경우, 'Pellow Island' 문자열을 포함하는 맵핑을 찾아 실제 맵핑 앞에 연결된 맵핑 이름을 갖는 단일 2진 문자열을 리턴합니다.

```
SELECT BLOB(MAP_NAME  || ':  ') || TOPOGRAPHIC_MAP
  FROM ONTARIO_SERIES_4
  WHERE TOPOGRAPIC_MAP LIKE BLOB('%Pellow Island%')
```

CEILING 또는 CEIL



스키마는 SYSFUN입니다.

인수(argument)보다 크거나 같은 최소의 정수 값을 리턴합니다.

인수는 내장형 숫자 유형이면 어느 것이나 가능합니다. 인수의 유형이 DECIMAL 또는 REAL일 경우, 함수가 처리할 수 있도록 인수는 배정밀도 부동 소수점 수로 변환됩니다. 인수 유형이 SMALLINT 또는 INTEGER일 경우, 인수 값이 리턴됩니다.

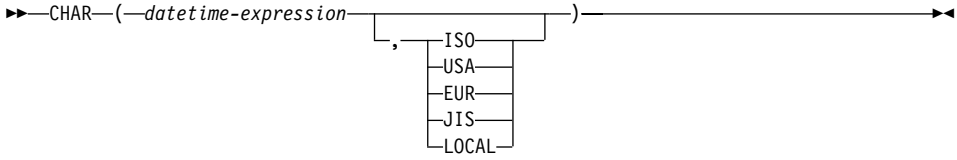
함수의 결과는 다음과 같습니다.

- 인수가 SMALLINT이면 SMALLINT입니다.
- 인수가 INTEGER이면 INTEGER입니다.
- 인수가 BIGINT이면 BIGINT입니다.
- 인수가 DECIMAL, REAL 또는 DOUBLE일 경우, DOUBLE입니다. 소수 왼쪽으로 15자리 이상의 십진 값은 DOUBLE로의 변환시 정밀도가 상실되므로 원하는 정수 값을 리턴하지 않습니다.

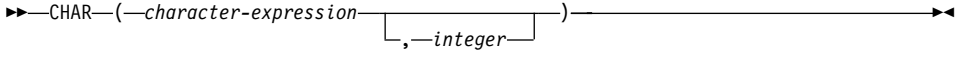
결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

CHAR

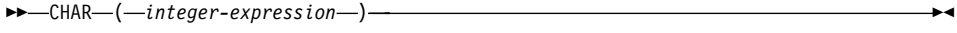
날짜시간 대 문자:



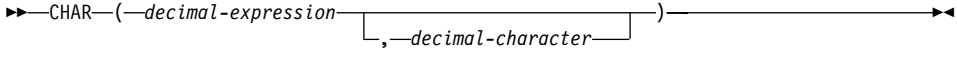
문자 대 문자:



정수 대 문자:



십진수 대 문자:



부동 소수점 대 문자:

스키마는 SYSIBM입니다. 그러나, CHAR(부동 소수점 표현식)에 대한 스키마는 SYSFUN입니다.

CHAR 함수는 다음의 문자열 표현을 리턴합니다.

- 첫번째 인수(augument)가 날짜, 시간 또는 시각인 경우, 날짜 시간값
- 첫번째 인수가 임의의 문자열 유형인 경우, 문자열 값
- 첫번째 인수가 SMALLINT, INTEGER 또는 BIGINT인 경우, 정수
- 첫번째 인수가 십진수인 경우, 십진수
- 첫번째 인수가 DOUBLE 또는 REAL일 경우, 배정밀도의 부동 소수점 수.

함수의 결과는 고정 길이 문자열입니다. 첫번째 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 첫번째 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

날짜시간 대 문자

날짜시간 표현식

다음의 세 가지 데이터 유형 중 하나인 표현식.

날짜 결과는 두 번째 인수에 의해 지정되는 형식으로 된 날짜의 문자열 표현. 결과의 길이는 10입니다. 두 번째 인수가 지정되었는데 그 값이 유효하지 않은 경우 오류가 발생합니다 (SQLSTATE 42703).

시간 결과는 두 번째 인수에 의해 지정되는 형식으로 된 시간의 문자열 표현. 결과의 길이는 8입니다. 두 번째 인수가 지정되었는데 그 값이 유효하지 않은 경우 오류가 발생합니다 (SQLSTATE 42703).

시간소인

두 번째 인수는 적용될 수 없으므로 지정하지 마십시오. SQLSTATE 42815의 결과는 시간소인의 문자열 표현입니다. 결과의 길이는 26입니다.

문자열의 코드 페이지는 응용프로그램 서버(AS)에 있는 데이터베이스의 코드 페이지입니다.

문자 대 문자

문자 표현식

CHAR, VARCHAR, LONG VARCHAR 또는 CLOB 데이터 유형인 값을 리턴하는 표현식.

정수

결과로 나오는 고정 길이 문자열에 대한 길이 속성. 값은 0과 254 사이여야 합니다.

문자 표현식의 길이가 결과의 길이 속성보다 작으면, 결과는 최대 결과 길이까지 공백이 채워집니다. 문자 표현식의 길이가 결과의 길이 속성보다 길

면 절단됩니다. 절단된 문자가 모두 공백이고 문자 표현식이 long 문자열(LONG VARCHAR 또는 CLOB)이 아니면, 경고가 리턴됩니다(SQLSTATE 01004).

정수 대 문자

정수-표현식

데이터 유형이 정수(SMALLINT, INTEGER 또는 BIGINT)인 값을 리턴하는 표현식.

결과는 SQL 정수 상수의 양식으로 된 인수의 문자열 표현입니다. 결과는 인수가 음수이면 음의 부호가 앞에 오는 인수의 값을 나타내는 유효 자릿수인 n개의 문자로 구성됩니다. 이것은 왼쪽으로 정렬됩니다.

- 첫번째 인수가 작은 정수인 경우:

  결과의 길이는 6입니다. 결과의 문자 수가 6자 미만이면, 결과는 6자가 되도록 오른쪽에 공백이 채워집니다.

- 첫번째 인수가 큰 정수인 경우:

  결과의 길이는 11입니다. 결과의 문자 수가 11자 미만이면, 결과는 11자가 되도록 오른쪽에 공백이 채워집니다.

- 첫번째 인수가 대 정수(big integer)인 경우

  결과의 길이는 20입니다. 결과의 문자 수가 20자 미만이면, 길이가 20자가 되도록 결과의 오른쪽에 공백이 채워집니다.

문자열의 코드 페이지는 응용프로그램 서버(AS)에 있는 데이터베이스의 코드 페이지입니다.

십진수 대 문자

십진수-표현식

소수 데이터 유형인 값을 리턴하는 표현식. 다른 정밀도와 자릿수를 원할 경우, 변경하기 위해 먼저 DECIMAL 스칼라 함수를 사용할 수 있습니다.

소수점-문자

결과 문자열에서 소수 자릿수를 분리하는 데 사용되는 1바이트 문자

상수를 지정합니다. 문자는 숫자, 더하기('+'), 빼기('-') 또는 공백이 될 수 없습니다 (SQLSTATE 42815). 기본값은 마침표('.') 입니다.

결과는 인수의 고정 길이 문자열 표현입니다. 결과에는 소수점 문자와 *p* 자리의 숫자가 포함됩니다. 여기서 *p*는 인수가 음수일 경우 음의 부호가 앞에 붙는 *십진수 표현식*의 정밀도입니다. 결과의 길이는 2+*p*입니다. 여기서, *p*는 *십진수 표현식*의 정밀도입니다. 이것은 양수 값에는 항상 하나의 뒷 공백이 포함됨을 의미합니다.

문자열의 코드 페이지는 응용프로그램 서버(AS)에 있는 데이터베이스의 코드 페이지입니다.

부동 소수점 대 문자

부동 소수점 표현식

부동 소수점 데이터 유형(DOUBLE 또는 REAL)인 값을 리턴하는 표현식.

결과는 부동 소수점 상수 형식의 인수에 대한 고정 길이 문자열 표현입니다. 결과는 길이 24입니다. 인수가 음수일 경우, 결과의 첫번째 문자는 마이너스 기호입니다. 그렇지 않으면, 첫번째 문자가 숫자입니다. 인수 값이 0일 경우, 결과는 0E0입니다. 그렇지 않으면, 결과에는 가수(mantissa)가 0이외의 단일 숫자와 마침표 및 일련의 숫자로 구성되는 것처럼, 인수의 값을 표현할 수 있는 최소 문자 수가 포함됩니다. 결과의 문자 수가 24자 미만일 경우, 길이가 24가 되도록 결과의 오른쪽에 공백이 채워집니다.

문자열의 코드 페이지는 응용프로그램 서버(AS)에 있는 데이터베이스의 코드 페이지입니다.

예:

- PRSTDATE 컬럼이 1988-12-25에 해당되는 값을 갖고 있다고 가정합니다.

```
CHAR(PRSTDATE, USA)
```

  결과 값은 '12/25/1988'입니다.

- 컬럼 STARTING에 17.12.30과 같은 내부 값이 있다고 가정할 경우, 호스트 변수 HOUR_DUR (decimal(6,0))은 값이 050000(즉, 5시간)인 시간 기간입니다.

```
CHAR(STARTING, USA)
```

결과 값은 '5:12 PM'입니다.

```
CHAR(STARTING + :HOUR_DUR, USA)
```

결과 값은 '10:12 PM'입니다.

- RECEIVED 컬럼(시각)이 PRSTDATE와 STARTING 컬럼의 조합에 해당되는 내부 값을 갖는다고 가정합니다.

```
CHAR(RECEIVED)
```

결과 값은 '1988-12-25-17.12.30.000000'입니다.

- CHAR 함수를 사용하여 고정 길이 문자를 입력한 후 EMPLOYEE 테이블의 LASTNAME 컬럼(VARCHAR(15)에 정의된 대로)에 대해 표시되는 문자 길이를 10자로 생성합니다.

```
SELECT CHAR(LASTNAME,10) FROM EMPLOYEE
```

10자보다 긴 길이(뒷 공백을 포함하여)의 LASTNAME을 갖고 있는 행에 대해, 값이 절단된다는 경고가 리턴됩니다.

- CHAR 함수를 사용하여 고정 길이 문자열로 EDLEVEL(smallint로 정의됨)의 값을 리턴합니다.

```
SELECT CHAR(EDLEVEL) FROM EMPLOYEE
```

EDLEVEL 18은 CHAR(6)값 '18 '(18 다음에 4개의 공백이 옴)로 리턴됩니다.

- STAFF에 정밀도 9, 스케일 2의 십진수로 정의된 SALARY 컬럼이 있다고 가정합시다. 현재 값은 18357.50이고 이것은 쉼표를 사용하여 십진수(18357,50)로 표시됩니다.

```
CHAR(SALARY, ',')
```

값 '00018357,50'이 리턴됩니다.

- 20000.25에서 뺀 동일한 SALARY 컬럼이 기본값과 함께 표시된다고 가정합니다.

```
CHAR(20000.25 - SALARY)
```

값 '-0001642.75'이 리턴됩니다.

- 호스트 변수 SEASONS_TICKETS가 정수 데이터 유형과 10000 값을 갖는다고 가정합니다.

```
CHAR(DECIMAL(:SEASONS_TICKETS,7,2))
```

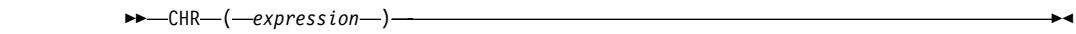
결과는 문자 값 '10000.00'입니다.

- 호스트 변수, DOUBLE_NUM이 2바이트 데이터 유형과 -987.654321E-35의 값을 갖는다고 가정합니다.

```
CHAR(:DOUBLE_NUM)
```

문자 값의 결과는 다음과 같습니다. '-9.87654321E-33 '. 결과 데이터 유형이 CHAR(24)이므로, 여기서 결과의 뒤 공백은 9입니다.

CHR



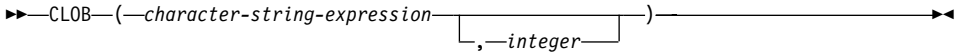
스키마는 SYSFUN입니다.

인수(argument)에 의해 지정된 ASCII 코드 값을 갖고 있는 문자를 리턴합니다.

인수는 INTEGER 또는 SMALLINT일 수 있습니다. 인수 값은 0과 255 사이어야 합니다. 그렇지 않으면, 리턴 값은 널(NULL)입니다.

함수의 결과는 CHAR(1)입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

CLOB



스키마는 SYSIBM입니다.

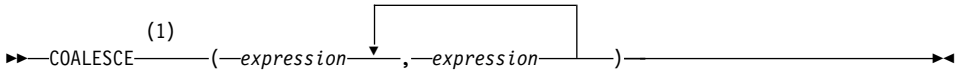
CLOB 함수는 문자열 유형의 CLOB 표현을 리턴합니다.

문자열 표현식
문자열인 값을 리턴하는 *표현식*.

정수
결과가 되는 CLOB 데이터 유형의 길이 속성을 지정하는 정수 값. 값은 0과 2 147 483 647 사이이어야 합니다. *정수*가 지정되지 않으면, 결과의 길이는 첫번째 인수의 길이와 같습니다.

함수의 결과는 CLOB입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

COALESCE



주:

1 VALUE는 COALESCE의 동의어입니다.

스키마는 SYSIBM입니다.

COALESCE는 널(NULL)이 아닌 첫번째 인수(argument)를 리턴합니다.

인수는 지정된 순서대로 평가되고, 함수의 결과는 널(NULL)이 아닌 첫번째 인수입니다. 모든 인수가 널(NULL)이 될 수 있는 경우에만 결과는 널(NULL)이 될 수 있고, 모든 인수가 널(NULL)인 경우에만 결과는 널(NULL)이 됩니다. 필요한 경우, 선택된 인수는 결과의 속성으로 변환됩니다.

인수는 호환 가능해야 합니다. 호환 가능한 데이터 유형과 결과의 속성에 대해서는 125 페이지의 『결과 데이터 유형 규칙』에서 참조하십시오. 내장 데이터 유형이거나 사용자 정의 데이터 유형입니다. [40]

예:

- DEPARTMENT 테이블에 있는 모든 행으로부터 모든 값을 선택할 경우, 부서 관리자(MGRNO)가 빠지면(즉, 널(NULL)이면), 'ABSENT' 값이 리턴됩니다.

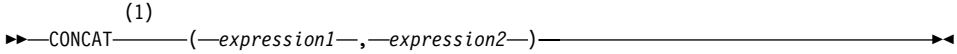
```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
  FROM DEPARTMENT
```

- EMPLOYEE 테이블에 있는 모든 행으로부터 사원 번호(EMPNO)와 급여(SALARY)를 선택할 경우, 급여가 빠지면(즉, 널(NULL)이면), 0 값이 리턴됩니다.

```
SELECT EMPNO, COALESCE(SALARY, 0)
  FROM EMPLOYEE
```

40. 사용자 정의 함수(UDF) 작성시, 이 함수를 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 호환성 있는 데이터 유형을 허용하므로, 사용자 정의 구별 유형을 지원하는 추가 시그니처를 작성할 필요가 없습니다.

CONCAT



주:

1 ||는 CONCAT의 동의어로 사용됩니다.

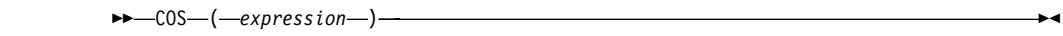
스키마는 SYSIBM입니다.

두 문자열 인수(argument)의 병합을 리턴합니다. 두 인수는 호환 가능한 유형이어야 합니다.

함수의 결과는 문자열입니다. 이 길이는 두 인수의 길이를 합한 것입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

184 페이지의 『병합 연산자가 있는 표현식』에서 자세한 설명을 참조하십시오.

COS



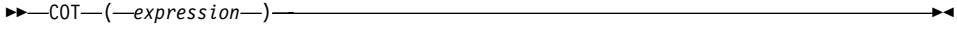
스키마는 SYSFUN입니다.

인수의 코사인을 리턴하며, 여기서 인수는 라디안으로 표시되는 각도입니다.

인수는 내장형 숫자 유형이면 어느 것이나 가능합니다. 이는 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 &dpfpn;입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

COT



스키마는 SYSFUN입니다.

인수(argument)의 코탄젠트를 리턴하며, 여기서 인수는 라디안으로 표시되는 각도입니다.

인수는 내장형 숫자 유형이면 어느 것이나 가능합니다. 이는 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 &dpfpn;입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

DATE

스키마는 SYSIBM입니다.

DATE 함수는 값에서 날짜(date)를 리턴합니다.

인수는 날짜, 시각, 3 652 059이하의 양수, 날짜나 시각의 유효한 문자열 표현 또는 CLOB도 LONG VARCHAR도 아닌 길이가 7인 문자열이어야 합니다.

인수가 길이 7의 문자열이면, 이것은 *yyyynnn* 형식으로 된 유효한 날짜를 표시해야 합니다. 여기서, *yyyy*는 년도를 나타내는 숫자이고, *nnn*은 그 해의 일(day)을 나타내는 001 - 366 사이의 숫자입니다.

함수의 결과는 날짜입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 값의 날짜 부분입니다.
- 인수가 숫자이면:
  - 결과는 1월 1일(0001) 이후의 *n*-1일이 되는 날짜입니다. 여기서, *n*은 숫자의 정수 부분입니다.
- 인수가 길이 7의 문자열인 경우:
  - 결과는 문자열로 표시되는 날짜입니다.

예:

- RECEIVED 컬럼(시각)이 '1988-12-25-17.12.30.000000'에 해당되는 내부 값을 갖고 있는 것으로 가정합니다.

  ```
  DATE(RECEIVED)
  ```

  결과는 '1988-12-25'의 내부 표현입니다.
- 다음 예의 결과는 '1988-12-25'의 내부 표현입니다.

```
DATE('1988-12-25')
```

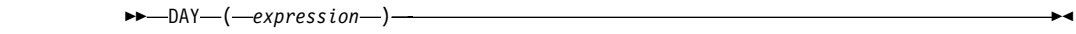
- 다음 예의 결과는 '1988-12-25'의 내부 표현입니다.

```
DATE('25.12.1988')
```

- 다음 예의 결과는 '0001-02-04'의 내부 표현입니다.

```
DATE(35)
```

DAY



스키마는 SYSIBM입니다.

DAY 함수는 값에서 일(day) 부분을 리턴합니다.

인수는 날짜, 시각, 날짜 기간, 시각 기간 또는 CLOB도 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 값의 일(day) 부분으로, 1 - 31 사이의 정수입니다.
- 인수가 날짜 기간이거나 시각 기간인 경우:
  - 결과는 값의 일(day) 부분으로, 1 - 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- PROJECT 테이블을 사용하여, 호스트 변수 END_DAY (smallint)를 WELD LINE PLANNING 프로젝트(PROJNAME)가 중지(PRENDATE)될 것으로 스케줄된 날짜로 설정합니다.

```
SELECT DAY(PRENDATE)
  INTO :END_DAY
               FROM PROJECT
  WHERE PROJNAME = 'WELD LINE PLANNING'
```

  샘플 테이블을 사용할 경우 END_DAY가 15로 설정됩니다.

- DATE1 컬럼(날짜)은 2000-03-15에 해당되는 내부 값을 갖고, DATE2 컬럼(날짜)은 1999-12-31에 해당되는 내부 값을 갖고 있다고 가정합니다.

```
DAY(DATE1 - DATE2)
```

  결과 값은 15입니다.

DAYNAME

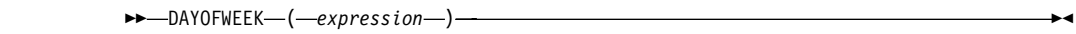
스키마는 SYSFUN입니다.

베이스가 시작될 때의 로케일에 근거하여 인수의 일자부분에 대한 요일 이름(예: 금요일)이 들어 있는 대소문자 혼합 문자열을 리턴합니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 VARCHAR(100)입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

DAYOFWEEK



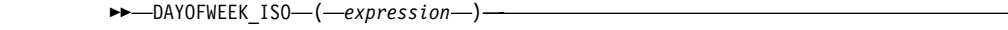
스키마는 SYSFUN입니다.

인수에 있는 요일을 1-7 범위내의 정수 값으로 리턴하며, 여기서 1은 일요일을 나타냅니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

DAYOFWEEK_ISO



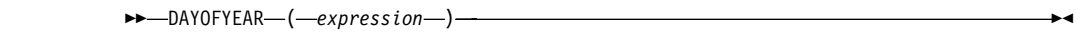
스키마는 SYSFUN입니다.

인수에 있는 요일을 1-7 범위내의 정수 값으로 리턴합니다. 여기서 1은 월요일을 나타냅니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

DAYOFYEAR



스키마는 SYSFUN입니다.

인수에 있는 연도의 날짜를 1-366 범위내의 정수 값으로 리턴합니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

DAYS

스키마는 SYSIBM입니다.

DAYS 함수는 날짜의 정수 표현을 리턴합니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

결과는 1월 1일(0001)에서 *D*까지의 일수보다 1이 큰 수입니다. 여기서, *D*는 DATE 함수가 인수에 적용된 경우 발생하는 날짜입니다.

예:

- PROJECT 테이블을 사용하여, 호스트 변수 EDUCATION_DAYS (int)를 프로젝트(PROJNO) 'IF2000'에 대해 예측된 경과 일 수(PRENDATE - PRSTDATE)로 설정합니다.

  ```
  SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
    INTO :EDUCATION_DAYS
    FROM PROJECT
    WHERE PROJNO = 'IF2000'
  ```

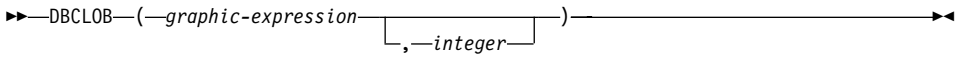
  샘플 테이블을 사용할 경우 EDUCATION_DAYS가 396으로 설정됩니다.

- PROJECT 테이블을 사용하여, 호스트 변수 TOTAL_DAYS (int)를 부서(DEPTNO) 'E21' 내의 모든 프로젝트에 대해 예측된 총 경과 일 수(PRENDATE - PRSTDATE)로 설정합니다.

  ```
  SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
    INTO :TOTAL_DAYS
    FROM PROJECT
    WHERE DEPTNO = 'E21'
  ```

  샘플 테이블을 사용할 경우 TOTAL_DAYS가 1584로 설정됩니다.

DBCLOB



스키마는 SYSIBM입니다.

DBCLOB 함수는 그래픽 문자열 유형의 DBCLOB 표현을 리턴합니다.

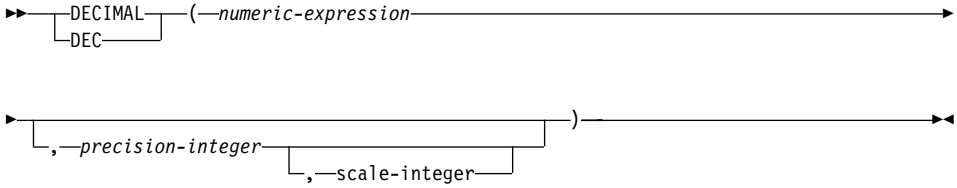
그래픽 표현식
그래픽 문자열인 값을 리턴하는 *표현식*

정수
결과가 되는 DBCLOB 데이터 유형의 길이 속성을 지정하는 정수 값. 값은 0과 1 073 741 823 사이여야 합니다. *정수*가 지정되지 않으면, 결과의 길이는 첫번째 인수의 길이와 같습니다.

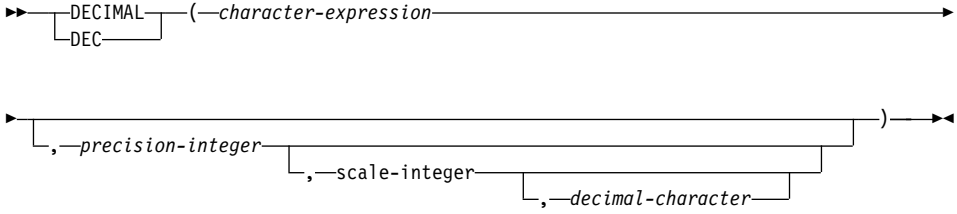
함수의 결과는 DBCLOB입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

DECIMAL

숫자 대 십진수:



문자 대 십진수:



스키마는 SYSIBM입니다.

DECIMAL 함수는 다음의 십진수 표현을 리턴합니다.

- 숫자
- 십진수의 문자열 표현
- 정수의 문자열 표현

함수의 결과는 정밀도가 *p*이고 스케일이 *s*인 십진수입니다. 여기서, *p*와 *s*는 두 번째 및 세 번째 인수입니다. 첫번째 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, 첫번째 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

숫자 대 십진수

숫자 표현식
임의의 숫자 데이터 유형인 값을 복귀하는 표현식.

정밀도-정수
값이 1 - 31 범위 사이 값을 갖고 있는 정수 상수.

*정밀도 정수*에 대한 기본값은 *숫자 표현식*의 데이터 유형에 따라 다릅니다.

- 부동 소수점 및 십진수의 경우에는 15
- 대 정수(big integer)의 경우에는 19
- 큰 정수의 경우에는 11
- 작은 정수의 경우에는 5

스케일-정수

범위가 0 - *정밀도-정수* 값인 정수 상수. 기본값은 0입니다.

결과는 정밀도가 *p*이고 스케일이 *s*인 십진수 컬럼이나 변수 첫번째 인수가 지정된 경우에 발생하는 것과 같은 수입니다. 여기서, *p*와 *s*는 두 번째 및 세 번째 인수입니다. 수의 정수 부분을 나타내는 데 필요한 유효 십진 자릿수가 *p*-*s*보다 크면 오류가 발생합니다.

문자 대 십진수

문자 표현식

문자 상수의 최대 길이(4 000바이트)보다 길지 않은 길이의 문자열인 값을 리턴시키는 *표현식*. 이것은 CLOB 또는 LONG VARCHAR 데이터 유형을 가질 수 없습니다. 앞뒤 공백은 문자열에서 삭제됩니다. 결과로 나오는 부속 문자열은 SQL 정수나 십진 상수를 형성하는 규칙에 따라야 합니다(SQLSTATE 22018).

*문자-표현식*은 필요에 따라 상수 *소수점-문자*의 코드 페이지에 일치되도록 데이터베이스 코드 페이지로 변환됩니다.

정밀도-정수

결과의 정밀도를 지정하는 범위 1 - 31 사이의 값을 갖는 정수 상수. 지정하지 않으면, 기본값은 15입니다.

스케일-정수

결과의 스케일을 지정하는 범위 0 - *정밀도-정수* 사이의 값을 갖는 정수 상수. 지정하지 않으면, 기본값은 0입니다.

소수점-문자

수의 정수 부분으로부터 *문자 표현식*의 십진 자릿수를 구분하는 데 사

용되는 1바이트 문자 상수를 지정합니다. 문자는 더하기('+'), 빼기('-') 또는 공백이 될 수 없으며, *문자-표현식*에 최대 한번 나타날 수 있습니다(SQLSTATE 42815).

결과는 정밀도가 *p*이고 스케일이 *s*인 소수입니다. 여기서, *p*와 *s*는 두 번째 및 세 번째 인수입니다. 디지트는 십진수의 오른쪽에 있는 자리수가 스케일 *s*보다 클 경우 끝이 절단됩니다. *문자 표현식*에서 십진수 왼쪽에 있는 유의 자리수(숫자의 정수 부분)가 *digit p-s*보다 클 경우 오류가 발생합니다(SQLSTATE 22003). 기본 소수점 문자는 *소수점 문자* 인수가 지정된 경우에 부속 문자열에서 유효하지 않습니다(SQLSTATE 22018).

예:

- EMPLOYEE 테이블에서 EDLEVEL 컬럼(데이터 유형 = SMALLINT)에 대한 선택 목록에서 DECIMAL 데이터 유형(정밀도가 5이고 스케일이 2)이 리턴하려면 DECIMAL 함수를 사용합니다. EMPNO 컬럼도 선택 목록에 나타나야 합니다.

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
                              FROM EMPLOYEE
```

- 호스트 변수 PERIOD의 유형을 INTEGER로 가정합니다. 그러면, 그 값을 날짜 기간으로 사용하기 위해 소수(8,0)로 "유형변환(cast)"해야 합니다.

```
SELECTPRSTDATE + DECIMAL(:PERIOD,8)
  FROM PROJECT
```

- SALARY 컬럼에 대한 갱신사항이 소수점 문자로 쉼표를 사용하는 문자열로서 창을 통한 입력이라고 가정합시다(예를 들어, 사용자는 21400,50을 입력함). 응용프로그램에 의해 유효성이 검사되면, 그것은 CHAR(10)로 정의되는 호스트 변수 newsalary에 지정됩니다.

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',')
  WHERE ID = :empid;
```

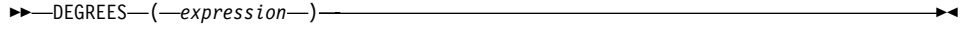
  newsalary의 값은 21400.50이 됩니다.

- 기본 소수점 문자(.)를 값에 추가합니다.

```
DECIMAL('21400,50', 9, 2, '.')
```

이것은 마침표(.)가 소수점 문자로 지정되었으나, 쉼표(,)가 분리 문자로 첫번째 인수에 나타났으므로 실패합니다.

DEGREES



스키마는 SYSFUN입니다.

라디안으로 표시된, 인수에서 변환한 차수(degree)의 수를 리턴합니다.

인수는 내장형 숫자 유형이면 어느 것이나 가능합니다. 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

DEREF

스키마는 SYSIBM입니다.

DEREF 함수는 인수의 목표 유형 인스턴스를 리턴합니다.

인수는 정의된 영역을 가진 참조 데이터 유형이 있는 값입니다(SQLSTATE 428DT).

결과의 정적 데이터 유형은 인수의 목표 유형입니다. 결과의 동적 데이터 유형은 인수의 목표 유형의 하위 유형입니다. 결과는 널(NULL)이 될 수 있습니다. *표현식*이 널(NULL) 값이거나 *표현식*이 목표 테이블에 일치하는 OID가 없는 참조인 경우, 결과는 널(NULL) 값이 됩니다.

결과는 참조의 목표 유형의 하위 유형 인스턴스입니다. 그 결과는 참조값에 일치하는 오브젝트 식별자를 가진 참조의 목표 뷰 또는 목표 테이블의 행을 찾음으로써 결과가 결정됩니다. 이 행의 유형은 결과의 동적 유형을 결정합니다. 결과 유형이 목표 테이블이나 목표 뷰의 하위 테이블 또는 하위 뷰의 행에 기초한 것일 수 있으므로, 명령문의 권한 부여 ID에는 목표 테이블 및 모든 하위 테이블 또는 목표 뷰 및 모든 하위 뷰에 대해 SELECT 특권이 있어야 합니다(SQLSTATE 42501).

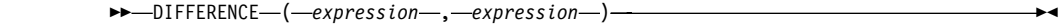
예:

EMPLOYEE가 유형 EMP의 테이블이고, 오브젝트 식별자 컬럼 이름이 EMPID라고 가정합시다. 이 때, 다음 조회는 EMPLOYEE 테이블과 하위 테이블의 각 행에 대해 유형 EMP 오브젝트 또는 해당되는 부속 유형을 리턴합니다. 이 조회에서는 EMPLOYEE와 해당되는 모든 하위 테이블에 대해 SELECT 특권을 요구합니다.

```
SELECT DEREF(EMPID) FROM EMPLOYEE
```

417 페이지의 『TYPE_NAME』에서 자세한 내용을 참조하십시오.

DIFFERENCE



스키마는 SYSFUN입니다.

문자열에 적용되는 SOUNDEX에 따라 두 문자열의 사운드 사이의 차이를 나타내는 0에서 4까지의 값을 리턴합니다. 값 4는 가능한 최상의 사운드입니다.

인수는 최대 4 000바이트의 CHAR 또는 VARCHAR인 문자열이 될 수 있습니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

예:

```
VALUES(DIFFERENCE('CONSTRAINT','CONSTANT'),SOUNDEX('CONSTRAINT'),SOUNDEX('CONSTANT')),
      (DIFFERENCE('CONSTRAINT','CONTRITE'),SOUNDEX('CONSTRAINT'),SOUNDEX('CONTRITE'))
```

이 예는 다음 결과를 리턴합니다.

```
1           2    3
----------- ---- ----
          4 C523 C523
          2 C523 C536
```

첫번째 행에서는 SOUNDEX와 동일한 결과를 보여주지만, 두 번째 행에서는 약간의 유사성만을 보여줍니다.

DIGITS

스키마는 SYSIBM입니다.

DIGITS 함수는 숫자의 문자열 표현을 리턴합니다.

인수는 유형이 SMALLINT, INTEGER, BIGINT 또는 DECIMAL인 값을 리턴하는 표현식이어야 합니다.

인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

함수의 결과는 스케일에 관계없이 인수의 절대값을 나타내는 고정 길이 문자열입니다. 결과에는 부호나 소수점 문자가 포함되지 않습니다. 대신, 필요에 따라 문자열을 채우기 위해 선행 0을 포함하는 숫자들로 구성됩니다. 문자열의 길이는 다음과 같습니다.

- 인수(argument)가 작은 정수이면 5
- 인수가 큰 정수이면 10
- 인수가 대 정수(big integer)이면 19
- 인수가 정밀도 *p*의 십진수이면 *p*

예:

- 10개 디지트를 포함하는 INTCOL이라고 하는 INTEGER 컬럼이 TABLEX 테이블에 포함되어 있다고 가정합니다. INTCOL 컬럼에 포함된 처음 네 개의 디지트의 고유한 모든 네 자리 조합을 나열합니다.

  ```
  SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
    FROM TABLEX
  ```

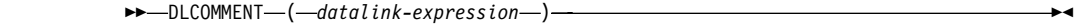
- COLUMNX가 DECIMAL(6,2) 데이터 유형을 갖고, 그 값 중 하나가 -6.28이라고 가정합니다. 그러면, 이 값에 대해

  ```
  DIGITS(COLUMNX)
  ```

  위 식은 값 '000628'을 리턴합니다.

결과는 이 길이만큼 문자열을 선행 0으로 채우는 길이 6(컬럼의 정밀도)의 문자열입니다. 결과에는 부호도 소수점도 나타나지 않습니다.

DLCOMMENT



스키마는 SYSIBM입니다.

DLCOMMENT 함수는 DATALINK 값으로부터 주석 값을 리턴합니다.

인수는 데이터 유형 DATALINK가 있는 값이 되는 표현식이어야 합니다.

함수의 결과는 VARCHAR(254)입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예:

- HOCKEY_GOALS 테이블의 ARTICLES 컬럼으로의 링크로부터 설명, 주석 및 날짜를 선택하도록 명령문을 준비합니다. 선택될 행들은 Richard 형제 중 하나(Maurice 또는 Henri)에 의해 점수가 계산되는 목표에 대한 행들입니다.

```
stmtvar = "SELECT DATE_OF_GOAL, DESCRIPTION, DLCOMMENT(ARTICLES)
             FROM HOCKEY_GOALS
             WHERE BY_PLAYER = 'Maurice Richard' OR BY_PLAYER = 'Henri Richard' ";
EXEC SQL PREPARE HOCKEY_STMT FROM :stmtvar;
```

- 스칼라 함수를 사용하여 TBLA 테이블에 있는 행의 COLA 컬럼으로 삽입된 DATALINK의 값이 다음과 같은 경우,

```
DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment')
```

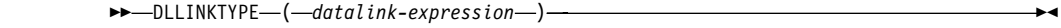
  그 값에 대해 작동되는 다음과 같은 함수는,

```
DLCOMMENT(COLA)
```

  다음 값을 리턴합니다.

```
A comment
```

DLLINKTYPE



스키마는 SYSIBM입니다.

DLLINKTYPE 함수는 DATALINK 값으로부터 링크 유형 값을 리턴합니다.

인수는 데이터 유형이 DATALINK인 값이 결과로 나오는 표현식이어야 합니다.

함수의 결과는 VARCHAR(4)입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예:

- 스칼라 함수를 사용하여 TBLA 테이블에 있는 행의 COLA 컬럼으로 삽입된 DATALINK의 값이 다음과 같은 경우,

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

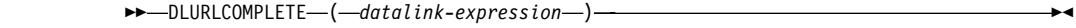
  그 값에 대해 작동되는 다음과 같은 함수는,

  ```
  DLLINKTYPE(COLA)
  ```

  다음 값을 리턴합니다.

  ```
  URL
  ```

DLURLCOMPLETE



스키마는 SYSIBM입니다.

DLURLCOMPLETE 함수는 링크 유형이 URL인 DATALINK 값으로부터 데이터 위치 속성을 리턴합니다. 적절한 경우 이 값에는 파일 액세스 토큰이 포함됩니다.

인수는 데이터 유형이 DATALINK인 값이 결과로 나오는 표현식이어야 합니다.

함수의 결과는 VARCHAR(254)입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

DATALINK 값에 주석만 포함되어 있으면, 리턴되는 결과는 길이가 0인 문자열입니다.

예:

- 스칼라 함수를 사용하여 TBLA 테이블에 있는 행의 COLA 컬럼으로 삽입된 DATALINK의 값이 다음과 같은 경우,

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

  그 값에 대해 작동되는 다음과 같은 함수는,

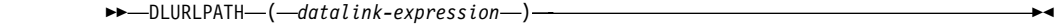
  ```
  DLURLCOMPLETE(COLA)
  ```

  다음 값을 리턴합니다.

  ```
  HTTP://DLFS.ALMADEN.IBM.COM/x/y/****************;a.b
  ```

  (여기서 ****************는 액세스 토큰)

DLURLPATH



스키마는 SYSIBM입니다.

DLURLPATH 함수는 링크 유형 URL과 함께 DATALINK 값으로부터 제공된 서버 내에서 파일 액세스에 필요한 파일 이름 및 경로를 리턴합니다. 적절한 경우 이 값에는 파일 액세스 토큰이 포함됩니다.

인수는 데이터 유형이 DATALINK인 값이 결과로 나오는 표현식이어야 합니다.

함수의 결과는 VARCHAR(254)입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

DATALINK 값에 주석만 포함되어 있으면, 리턴되는 결과는 길이가 0인 문자열입니다.

예:

- 스칼라 함수를 사용하여 TBLA 테이블에 있는 행의 COLA 컬럼으로 삽입된 DATALINK의 값이 다음과 같은 경우,

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

  그 값에 대해 작동되는 다음과 같은 함수는,

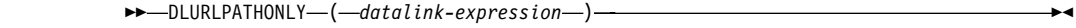
  ```
  DLURLPATH(COLA)
  ```

  다음 값을 리턴합니다.

  ```
  /x/y/****************;a.b
  ```

  (여기서 **********는 액세스 토큰)

DLURLPATHONLY



스키마는 SYSIBM입니다.

DLURLPATHONLY 함수는 링크 유형 URL과 함께 DATALINK 값으로부터 제공된 서버 내에서 파일 액세스에 필요한 파일 이름 및 경로를 리턴합니다 NEVER를 리턴한 값에는 파일 액세스 토큰이 포함됩니다.

인수는 데이터 유형이 DATALINK인 값이 결과로 나오는 표현식이어야 합니다.

함수의 결과는 VARCHAR(254)입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

DATALINK 값에 주석만 포함되어 있으면, 리턴되는 결과는 길이가 0인 문자열입니다.

예:

- 스칼라 함수를 사용하여 TBLA 테이블에 있는 행의 COLA 컬럼으로 삽입된 DATALINK의 값이 다음과 같은 경우,

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

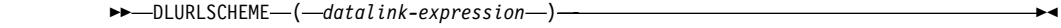
  그 값에 대해 작동되는 다음과 같은 함수는,

  ```
  DLURLPATHONLY(COLA)
  ```

  다음 값을 리턴합니다.

  ```
  /x/y/a.b
  ```

DLURLSCHEME



스키마는 SYSIBM입니다.

DLURLSCHEME 함수는 링크 유형 URL과 함께 DATALINK 값으로부터 체계를 리턴합니다. 값은 항상 대문자가 됩니다.

인수는 데이터 유형이 DATALINK인 값이 결과로 나오는 표현식이어야 합니다.

함수의 결과는 VARCHAR(20)입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

DATALINK 값에 주석만 포함되어 있으면, 리턴되는 결과는 길이가 0인 문자열입니다.

예:

- 스칼라 함수를 사용하여 TBLA 테이블에 있는 행의 COLA 컬럼으로 삽입된 DATALINK의 값이 다음과 같은 경우,

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

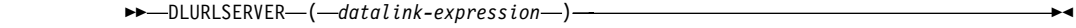
  그 값에 대해 작동되는 다음과 같은 함수는,

  ```
  DLURLSCHEME(COLA)
  ```

  다음 값을 리턴합니다.

  ```
  HTTP
  ```

DLURLSERVER



스키마는 SYSIBM입니다.

DLURLSERVER 함수는 링크 유형 URL과 함께 DATALINK 값으로부터 파일 서버를 리턴합니다. 값은 항상 대문자가 됩니다.

인수는 데이터 유형이 DATALINK인 값이 결과로 나오는 표현식이어야 합니다.

함수의 결과는 VARCHAR(254)입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

DATALINK 값에 주석만 포함되어 있으면, 리턴되는 결과는 길이가 0인 문자열입니다.

예:

- 스칼라 함수를 사용하여 TBLA 테이블에 있는 행의 COLA 컬럼으로 삽입된 DATALINK의 값이 다음과 같은 경우,

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

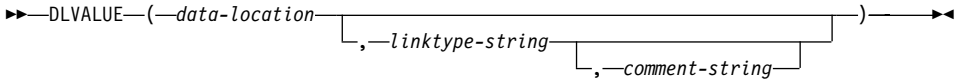
  그 값에 대해 작동되는 다음과 같은 함수는,

  ```
  DLURLSERVER(COLA)
  ```

  다음 값을 리턴합니다.

  ```
  DLFS.ALMADEN.IBM.COM
  ```

DLVALUE



스키마는 SYSIBM입니다.

DLVALUE 함수는 DATALINK 값을 리턴합니다. 함수가 UPDATE문의 SET절 오른쪽에 있거나 INSERT문의 VALUES절에 있는 경우, 일반적으로 파일로의 링크도 작성합니다. 그러나, 주석만 지정되는 경우(데이터 위치가 제로 길이 문자열인 경우), DATALINK 값은 링크 속성이 빈 채로 작성되므로 파일 링크가 이루어지지 않습니다.

데이터-위치
: 링크 유형이 URL인 경우, 이것은 완전한 URL 값이 들어 있는 가변 길이 문자열을 산출하는 표현식입니다.

링크유형-문자열
: DATALINK 값의 링크 유형을 지정하는 선택적 VARCHAR 표현식. 유일한 유효값은 'URL'(SQLSTATE 428D1)입니다.

주석-문자열
: 주석이나 추가 위치 정보를 제공하는 선택적 VARCHAR(254) 값.

함수 결과는 DATALINK 값입니다. DLVALUE 함수의 인수 중 하나가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. *data-location*이 널(NULL)이면, 결과는 널(NULL) 값입니다.

이 함수를 사용하여 DATALINK 값을 정의할 때 값 목표의 최대 길이를 고려하십시오. 예를 들어, 컬럼이 DATALINK(200)로서 정의되는 경우, *데이터-위치*와 *주석*을 합한 최대 길이는 200바이트입니다.

예:

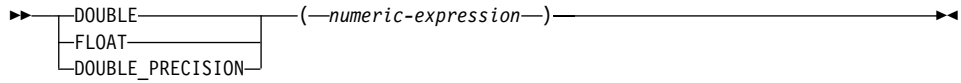
- 테이블에 한 행을 삽입합니다. 첫번째 두 링크에 대한 URL 값은 url_article과 url_snapshot 변수에 포함됩니다. url_snapshot_comment 변수에는 스냅샷 링

크를 수반하는 주석이 포함됩니다. 아직까지는 url_movie_comment 변수에 있는 주석인 movie에 대한 링크는 없습니다.

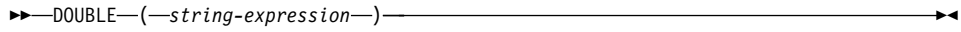
```
EXEC SQL  INSERT INTO HOCKEY_GOALS
            VALUES('Maurice Richard',
                   'Montreal Canadien',
                   '?',
                   'Boston Bruins,
                   '1952-04-24',
                   'Winning goal in game 7 of Stanley Cup final',
                    DLVALUE(:url_article),
                    DLVALUE(:url_snapshot, 'URL', :url_snapshot_comment),
                    DLVALUE('', 'URL', :url_movie_comment) );
```

DOUBLE

숫자 대 배수



문자열 대 배수:



스키마는 SYSIBM입니다. 그러나, DOUBLE(*문자열 표현식*)에 대한 스키마는 SYSFUN입니다.

DOUBLE 함수는 다음에 해당하는 부동 소수점 숫자를 리턴합니다.

- 인수가 숫자 표현식일 경우, 숫자
- 인수가 문자열 표현식일 경우, 숫자에 대한 문자열 표현

숫자 대 배수

숫자 표현식

인수는 내장 숫자 데이터 유형값을 리턴하는 표현식입니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

결과는 인수가 배정밀도 부동 소수점 컬럼이나 변수에 지정된 경우에 발생하는 것과 같은 수입니다.

문자열 대 배수

문자열 표현식

인수의 유형은 숫자 상수 형태의 CHAR 또는 VARCHAR입니다. 인수의 선행 및 뒷 공백은 무시됩니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

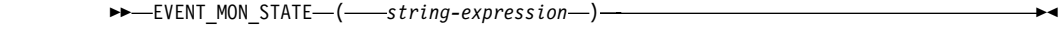
결과는 문자열이 상수로 간주되어 배정밀도의 부동 소수점 컬럼이나 변수에 지정될 경우 발생하는 수와 동일합니다.

예:

EMPLOYEE 테이블을 사용하여, 수당이 0이 아닌 사원들의 수당에 대한 급여 비율을 찾습니다. 관련된 컬럼(SALARY 및 COMM)은 DECIMAL 데이터 유형을 갖습니다. 범위 밖의 결과가 발생할 수 있는 가능성을 없애기 위해, 부동 소수점에서 나눗셈이 처리되도록 DOUBLE이 SALARY에 적용됩니다.

```
SELECT EMPNO, DOUBLE(SALARY)/COMM
                              FROM EMPLOYEE
  WHERE COMM > 0
```

EVENT_MON_STATE



스키마는 SYSIBM입니다.

EVENT_MON_STATE 함수는 이벤트 모니터의 현재 상태를 리턴합니다.

인수는 결과 데이터 유형이 CHAR 또는 VARCHAR이고, 이벤트 모니터의 이름인 값을 갖는 문자열 표현식입니다. 명명된 이벤트 모니터가 SYSCAT.EVENTMONITORS 카탈로그 테이블에 존재하지 않으면, SQLSTATE 42704가 리턴됩니다.

결과는 다음 값 중 하나의 정수입니다.

•

0 이벤트 모니터는 비활성 상태입니다.

1 이벤트 모니터는 활성 상태입니다.

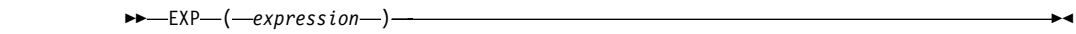
인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예:

• 다음 예는 정의된 모든 이벤트 모니터를 선택하고, 각 모니터가 활동중인지, 그렇지 않은 지를 나타냅니다.

```
SELECT EVMONNAME,
   CASE
      WHEN EVENT_MON_STATE(EVMONNAME) = 0 THEN 'Inactive'
      WHEN EVENT_MON_STATE(EVMONNAME) = 1  THEN 'Active'
   END
FROM SYSCAT.EVENTMONITORS
```

EXP



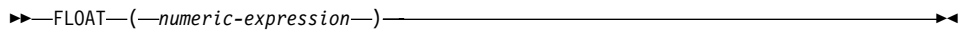
스키마는 SYSFUN입니다.

인수의 지수 함수(exponential function)를 리턴합니다.

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 이는 함수가 처리할 수 있도록 &dpfpn;로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

FLOAT

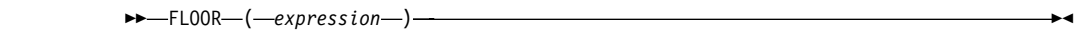


스키마는 SYSIBM입니다.

FLOAT 함수는 숫자의 부동 소수점 표현을 리턴합니다.

FLOAT는 DOUBLE의 동의어입니다. 330 페이지의 『DOUBLE』에서 자세한 내용을 참조하십시오.

FLOOR



스키마는 SYSFUN입니다.

인수보다 작거나 같은 최대의 정수 값을 리턴합니다.

인수는 내장형 숫자 유형이면 어느 것이나 가능합니다. 인수의 유형이 DECIMAL 또는 REAL일 경우, 함수가 처리할 수 있도록 인수는 배정밀도 부동 소수점 수로 변환됩니다. 인수 유형이 SMALLINT, INTEGER 또는 BIGINT이면, 인수 값이 리턴됩니다.

함수의 결과는 다음과 같습니다.

- 인수가 SMALLINT이면 SMALLINT입니다.
- 인수가 INTEGER이면 INTEGER입니다.
- 인수가 BIGINT이면 BIGINT입니다.
- 인수가 DOUBLE, DECIMAL 또는 REAL일 경우, DOUBLE입니다. 소수 왼쪽으로 15자리 이상의 십진 값은 DOUBLE로의 변환시 정밀도가 상실되므로 원하는 정수 값을 리턴하지 않습니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

GENERATE_UNIQUE

스키마는 SYSIBM입니다.

GENERATE_UNIQUE 함수는 함수의 다른 실행과 비교되는 고유한 13바이트 길이의 비트 데이터 문자열을 리턴합니다(CHAR(13) FOR BIT DATA). [41] 이 함수는 결정되지 않는 함수로 정의됩니다.

이 함수에 대한 인수가 없습니다(공백 괄호에 값을 지정해야 합니다.).

함수의 결과는 세계 표준시(UTC)와 함수가 처리된 파티션 번호를 포함하는 고유한 값입니다. 결과는 널(NULL)이 될 수 없습니다.

테이블에 고유한 값을 제공하기 위해 이 함수의 결과를 사용할 수 있습니다. 연속되는 각 값은 이전 값보다 크므로, 테이블 내에서 사용할 수 있는 일련의 순서를 제공합니다. 값에는 여러 파티션에 걸쳐 파티션된 테이블도 어떤 순서로 고유한 값을 갖도록 함수가 실행된 파티션 번호가 포함됩니다. 함수가 실행된 시간에 따라 순서가 설정됩니다.

이 함수는 여러 개의 행 삽입 명령문이나 fullselect를 갖는 삽입 명령문에 대해 고유 값이 생성된다는 점에서 특수 레지스터를 사용하는 것과는 다릅니다.

이 함수의 결과 가운데 한 부분인 시각 값은 GENERATE_UNIQUE의 결과를 인수로 한 상태에서 TIMESTAMP 스칼라 함수를 사용하여 정할 수 있습니다.

예:

- 각 행에 대해 고유한 컬럼을 포함하는 테이블을 작성하십시오. GENERATE_UNIQUE 함수를 사용하여 이 컬럼에 값을 지정하십시오. UNIQUE_ID 컬럼에는 컬럼을 비트 데이터 문자열로 나타내기 위해 "FOR BIT DATA"가 지정되어 있습니다.

41. 시스템 시계는 함수가 실행하고 있는 파티션 번호와 함께 내부 세계 표준시(UTC) 시간소인을 생성하는 데 사용됩니다. 실제 시스템 시계를 뒤로 조정하면 값이 중복될 수 있습니다.

```
CREATE TABLE EMP_UPDATE
(UNIQUE_ID CHAR(13) FOR BIT DATA,
                            EMPNO CHAR(6),
                            TEXT  VARCHAR(1000))

INSERT INTO EMP_UPDATE
          VALUES (GENERATE_UNIQUE(), '000020', 'Update entry...'),
          (GENERATE_UNIQUE(), '000050', 'Update entry...')
```

GENERATE_UNIQUE를 사용하여 UNIQUE_ID 컬럼을 설정하는 경우, 이 테이블의 각 행은 고유한 식별자를 갖게 됩니다. 이는 테이블에 트리거를 도입하여 수행할 수 있습니다.

```
CREATE TRIGGER EMP_UPDATE_UNIQUE
NO CASCADE BEFORE INSERT ON EMP_UPDATE
REFERENCING NEW AS NEW_UPD
       FOR EACH ROW MODE DB2SQL
SET NEW_UPD.UNIQUE_ID = GENERATE_UNIQUE()
```

이 트리거가 정의되면, 다음과 같이 첫번째 컬럼 없이 이전의 INSERT문을 실행할 수 있습니다.

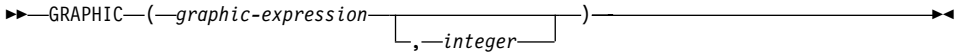
```
INSERT INTO EMP_UPDATE (EMPNO, TEXT)
           VALUES ('000020', 'Update entry 1...'),
           ('000050', '항목 2 갱신...')
```

다음을 사용하여 EMP_UPDATE에 행이 추가된 시간에 대한 시각(UTC에서)을 리턴할 수 있습니다.

```
       SELECT TIMESTAMP (UNIQUE_ID), EMPNO, TEXT  FROM EMP_UPDATE
```

그러므로, 행이 삽입된 시간을 레코드하기 위해 테이블에 시각 컬럼을 가질 필요가 없습니다.

GRAPHIC



스키마는 SYSIBM입니다.

GRAPHIC 함수는 그래픽 문자열 유형의 GRAPHIC 표현을 리턴합니다.

그래픽 표현식

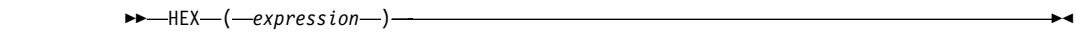
그래픽 문자열인 값을 리턴하는 *표현식*

정수

결과가 되는 GRAPHIC 데이터 유형의 길이 속성을 지정하는 정수 값. 값은 1과 127 사이여야 합니다. *정수*가 지정되지 않으면, 결과의 길이는 첫번째 인수의 길이와 같습니다.

함수의 결과는 GRAPHIC입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

HEX



스키마는 SYSIBM입니다.

HEX 함수는 문자열과 같은 값의 16진 표현을 리턴합니다.

인수는 최대 길이가 16 336 바이트인 내장 데이터 유형 값인 표현식일 수 있습니다.

함수의 결과는 문자열입니다. 인수가 널(NULL)이 될 수 있는 경우, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

코드 페이지는 데이터베이스 코드 페이지입니다.

결과는 16진수의 문자열입니다. 처음 두 디지트는 인수의 첫 바이트를 나타내고, 다음 두 디지트는 인수 두 번째 바이트를 나타냅니다. 인수가 날짜시간 값이거나 숫자 값이면, 결과는 인수의 내부 양식으로 된 16진 표현입니다. 리턴되는 16진 표현은 함수가 실행되는 응용프로그램 서버(AS)에 따라 다를 수 있습니다. 차이점이 명백한 경우는 다음과 같습니다.

- HEX 함수가 EBCDIC 서버가 있는 ASCII 클라이언트나 ASCII 서버가 있는 EBCDIC 클라이언트에서 수행될 경우의 문자열 인수입니다.
- HEX 함수가 수행될 때의 숫자 인수(일부 경우에서)입니다. 여기서, 클라이언트 및 서버 시스템에는 숫자값에 대해 순서화된 다른 바이트가 있습니다.

결과의 유형과 길이는 문자열 인수의 유형 및 길이에 따라 다양합니다.

- 문자열
  - 127보다 길지 않은 고정 길이
    - 결과는 인수의 정의된 길이의 두 배인 고정 길이의 문자열입니다.
  - 127보다 긴 고정 길이
    - 결과는 인수의 정의된 길이의 두 배인 가변 길이의 문자열입니다.
  - 가변 길이

  - 결과는 인수의 정의된 최대 길이의 두 배인 최대 길이를 갖는 가변 길이의 문자열입니다.

- 그래픽 문자열
  - 63보다 길지 않은 고정 길이
    - 결과는 인수의 정의된 길이의 네 배인 고정 길이의 문자열입니다.
- 63보다 긴 고정 길이
  - 결과는 인수의 정의된 길이의 네 배인 가변 길이의 문자열입니다.
- 가변 길이
  - 결과는 인수의 정의된 최대 길이의 네 배인 최대 길이를 갖는 가변 길이의 문자열입니다.

예:

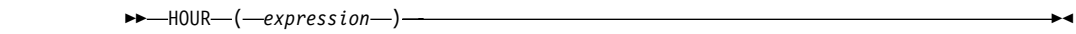
다음 예에서 AIX 응용프로그램 서버(AS)에 대해 DB2를 사용한다고 가정합시다.

- DEPARTMENT 테이블을 사용하여, 호스트 변수 HEX_MGRNO (char(12))를 'PLANNING' 부서(DEPTNAME)에 대한 관리자 번호(MGRNO)의 16진수 표시로 설정합니다.

```
SELECT HEX(MGRNO )
  INTO :HEX_MGRNO
  FROM DEPARTMENT
  WHERE DEPTNAME = 'PLANNING'
```

  샘플 테이블을 사용할 때 HEX_MGRNO가 '303030303230'으로 설정됩니다 (문자 값은 '000020'입니다).
- COL_1은 데이터 유형이 char(1)이고 값이 'B'인 컬럼으로 가정합니다. 문자 'B'의 16진 표현은 X'42'입니다. HEX(COL_1)는 두 자리 문자열 '42'를 리턴합니다.
- COL_3은 데이터 유형이 decimal(6,2)이고 값이 40.1인 컬럼으로 가정합니다. 문자열 '0004010C'는 HEX 함수를 십진수 40.0의 내부 표현에 적용한 결과입니다.

HOUR



스키마는 SYSIBM입니다.

HOUR 함수는 값의 시간 부분을 리턴합니다.

인수는 시간, 시각, 시간 기간, 시각 기간 또는 CLOB도 LONG VARCHAR도 아닌 시간이나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

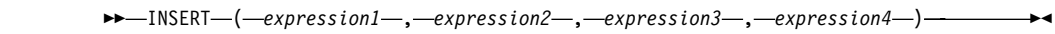
- 인수가 시간, 시간소인 또는 시간이나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 값에서 0과 24 사이의 시간 부분입니다.
- 인수가 시간 기간이거나 시각 기간인 경우:
  - 결과는 값의 시간 부분으로, 1 - 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

CL_SCHED 샘플 테이블을 사용하여, 정오에 시작하는 모든 수업을 선택합니다.

```
SELECT * FROM CL_SCHED
  WHERE HOUR(STARTING) BETWEEN 12 AND 17
```

INSERT



스키마는 SYSFUN입니다.

*표현식2*로 시작하는 *표현식1*에서 *표현식3*바이트가 삭제되고, *표현식2*로 시작하는 *표현식1*에 *표현식4*가 삽입된 문자열을 리턴합니다. 결과 문자열의 길이가 리턴 유형에 대한 최대값을 초과할 경우, 오류가 발생합니다(SQLSTATE 38552).

첫번째 인수는 문자열 또는 2진 문자열 유형입니다. 두 번째와 세번째 인수는 데이터 유형이 SMALLINI 또는 INTEGER인 숫자 값이어야 합니다. 첫번째 인수가 문자열일 경우, 네 번째 인수도 역시 문자열이어야 합니다. 첫번째 인수가 2진 문자열일 경우, 네 번째 인수도 2진 문자열이어야 합니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB 또는 2진 문자열의 최대 길이는 1 048 576바이트입니다. 첫번째와 네 번째 인수의 경우, CHAR은 VARCHAR로 변환되고 LONG VARCHAR는 CLOB(1M)로 변환됩니다. 그리고, 두 번째와 세 번째 인수 SMALLINT는 함수에서 처리되도록 INTEGER로 변환됩니다.

결과는 다음과 같이 인수 유형에 기초합니다.

- 첫번째와 네번째 인수가 모두 VARCHAR(4 000바이트 이하) 또는 CHAR이면 VARCHAR(4000)입니다.
- 첫번째 또는 네번째 인수가 CLOB 또는 LONG VARCHAR이면 CLOB(1M)입니다.
- 첫번째와 네번째 인수가 모두 BLOB이면 BLOB(1M)입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

예:

- 둘다 세번째 문자에서 시작하면 한 문자를 단어 'DINING'에서 삭제하고, 'VID'를 삽입하십시오.

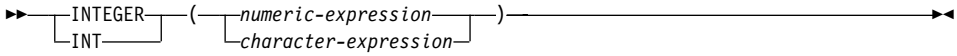
```
VALUES CHAR(INSERT('DINING', 3, 1, 'VID'), 10)
```

  이 예는 다음 결과를 리턴합니다.

```
1
----------
DIVIDING
```

언급한 것처럼, INSERT 함수의 출력은 VARCHAR(4000)입니다. 위의 예에서 함수 CHAR는 INSERT의 출력을 10바이트로 제한하는 데 사용됩니다. 특수 문자열의 시작 위치는 LOCATE를 사용하여 찾을 수 있습니다. 353 페이지의 『LOCATE』에서 좀더 자세한 정보를 참조하십시오.

INTEGER



스키마는 SYSIBM입니다.

INTEGER 함수는 정수 상수 형식의 숫자 또는 문자에 대해 정수 표현을 리턴합니다.

숫자 표현식

임의의 내장 숫자 데이터 유형값을 리턴시키는 표현식.

인수가 *숫자 표현식*이면, 결과는 인수가 큰 정수 컬럼 또는 변수에 지정되었을 경우에 발행한 것과 동일한 수입니다. 인수의 정수 부분이 정수 범위 내에 있지 않으면, 오류가 발생합니다. 인수의 소수 부분이 있으면, 절단됩니다.

문자 표현식

문자 상수의 최대 길이보다 작은 길이를 갖는 문자열 값을 리턴하는 표현식입니다. 전후 공백은 제거되고 결과 문자열은 SQL 정수 상수를 형성하는 규칙에 따라야 합니다(SQLSTATE 22018). 문자열은 long 문자열이 될 수 없습니다.

인수가 *문자 표현식*이면, 결과는 해당 정수 상수가 큰 정수 컬럼이나 변수에 지정되었을 때 발생하는 것과 같은 수가 됩니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

예:

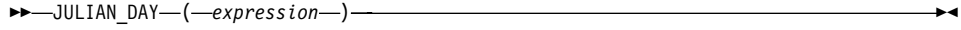
- EMPLOYEE 테이블을 사용하여, 교육 레벨(EDLEVEL)별로 나눠진 급여(SALARY)를 포함하는 목록을 선택합니다. 계산에서 소수는 절단됩니다. 목록에는 계산과 사원 번호에 사용되는 값도 포함되어야 합니다. 목록은 계산된 값이 내림차순으로 되어 있어야 합니다.

```
SELECT INTEGER(SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
                                  FROM EMPLOYEE
  ORDER BY 1 DESC
```

- EMPLOYEE 테이블을 사용하여, 응용프로그램에서의 추가 처리를 위해 정수 양식의 EMPNO 컬럼을 선택합니다.

```
SELECT INTEGER(EMPNO) FROM EMPLOYEE
```

JULIAN_DAY



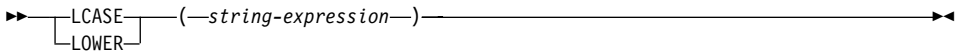
스키마는 SYSFUN입니다.

B.C 4712년 1월 1일(율리우스력의 시작)에서 인수에 지정된 날짜 값까지의 일 수를 나타내는 정수 값을 리턴합니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

LCASE 또는 LOWER



스키마는 SYSIBM입니다.[42]

LCASE 또는 LOWER 함수는 모든 SBCS 문자가 소문자로 변환된 문자열을 리턴합니다. 즉, A-Z 문자들이 a-z 문자들로 변환되고, 발음 구별 부호가 있는 문자는 이제 상응하는 소문자(있다면)로 변환될 것입니다. 예를 들어, 코드 페이지 850에서 É는 é로 맵핑됩니다. 모든 문자가 변환되지는 않으므로, LCASE(UCASE(*문자열 표현식*))가 꼭 LCASE(*문자열 표현식*)와 동일한 결과를 리턴할 필요는 없습니다.

인수는 그 값이 CHAR 또는 VARCHAR 데이터 유형인 표현식이어야 합니다.

함수의 결과는 인수와 동일한 데이터 유형 및 길이 속성을 가집니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

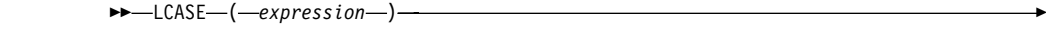
예: EMPLOYEE 테이블에서 JOB의 값에 있는 문자가 반드시 소문자로 리턴되게 하십시오.

```
SELECT LCASE(JOB)
  FROM EMPLOYEE WHERE EMPNO = '000020';
```

결과는 값 'manager'입니다.

42. 이 함수의 SYSFUN 버전이 LONG VARCHAR 및 CLOB 인수에 대한 지원과 계속 사용 가능합니다. 348 페이지의 『LCASE(SYSFUN 스키마)』에서 자세한 내용을 참조하십시오.

LCASE(SYSFUN 스키마)



스키마는 SYSFUN입니다.

A-Z까지의 모든 문자가 소문자(a-z)로 변환된 문자열을 리턴합니다(판독 기호가 붙은 문자는 변환되지 않습니다.). 그러므로, LCASE(UCASE(문자열))가 LCASE(문자열)와 반드시 동일한 결과를 리턴하는 것은 아닙니다.

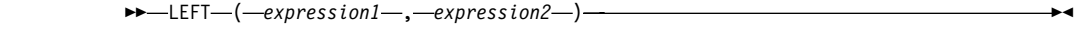
인수는 내장형 문자열 유형이면 어느 것이나 가능합니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB의 최대 길이는 1 048 576바이트입니다.

함수의 결과는 다음과 같습니다.

- 인수가 VARCHAR(4 000 이하) 또는 CHAR이면 VARCHAR(4000)입니다.
- 인수가 CLOB LONG 또는 VARCHAR이면 CLOB(1M)입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

LEFT



스키마는 SYSFUN입니다.

*표현식1*의 가장 왼쪽 *표현식2* 바이트로 구성되는 문자열을 리턴합니다. *표현식1* 값은 *표현식1*의 지정된 부속 문자열이 항상 존재하도록 효율적으로 필요한 만큼의 공백으로 오른쪽이 채워집니다.

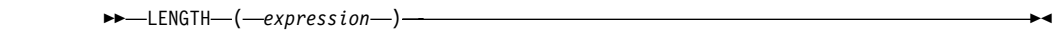
첫번째 인수는 문자열 또는 2진 문자열 유형입니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB 또는 2진 문자열의 최대 길이는 1 048 576바이트입니다. 두 번째 인수는 INTEGER 또는 SMALLINT 데이터 유형이어야 합니다.

함수의 결과는 다음과 같습니다.

- 인수가 VARCHAR(4 000 이하) 또는 CHAR이면 VARCHAR(4000)입니다.
- 인수가 CLOB LONG 또는 VARCHAR이면 CLOB(1M)입니다.
- 인수가 BLOB이면 BLOB(1M)입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

LENGTH



스키마는 SYSIBM입니다.

LENGTH 함수는 값의 길이를 리턴합니다.

인수는 내장 데이터 유형값을 리턴시키는 표현식일 수 있습니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

결과는 인수의 길이입니다. 길이에는 널(NULL) 값을 허용하는 컬럼 인수의 널(NULL) 표시기 바이트를 포함하지 않습니다. 문자열의 길이에는 공백은 포함되나 가변 길이 문자열의 길이 제어 필드는 포함되지 않습니다. 가변 길이 문자열의 길이는 실제 길이이나, 최대 길이는 아닙니다.

그래픽 문자열의 길이는 DBCS 문자 수입니다. 모든 다른 값들의 길이는 값을 표시하는 데 사용되는 바이트 수입니다.

- 작은 정수에 대해 2
- 큰 정수에 대해 4
- 정밀도가 *p*인 십진수에 대해 (*p*/2)+1
- 2진 문자열에 대해 문자열의 길이
- 문자열에 대해 문자열의 길이
- 단일 정밀도 부동 소수점에 대해 4
- 배정밀도 부동 소수점에 대해 8
- 날짜에 대해 4
- 시간에 대해 3
- 시각에 대해 10

예:

- ADDRESS 호스트 변수가 '895 Don Mills Road'의 값인 가변 길이 문자열이라고 가정합니다.

```
LENGTH(:ADDRESS)
```

값 18을 리턴합니다.

- START_DATE가 유형이 DATE인 컬럼으로 가정합니다.

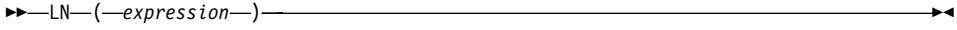
```
LENGTH(START_DATE)
```

값 4를 리턴합니다.

- 다음 예는 값 10을 리턴합니다.

```
LENGTH(CHAR(START_DATE, EUR))
```

LN



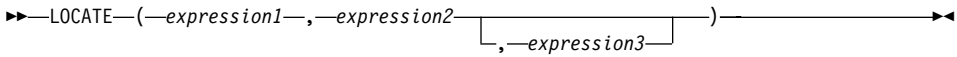
스키마는 SYSFUN입니다.

인수의 자연 로그를 리턴합니다(LOG와 동일).

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 &dpfpn;입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

LOCATE



스키마는 SYSFUN입니다.

표현식2 내에서 *표현식1*이 처음 발생한 시작 위치를 리턴합니다. 선택적 *표현식3*이 지정되면, 검색이 시작되는 *표현식2*의 문자 위치를 나타냅니다. *표현식1*이 *표현식2*에 없을 경우, 값 0이 리턴됩니다.

첫번째 인수가 문자열인 경우, 두 번째 인수도 문자열이어야 합니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB의 최대 길이는 1 048 576바이트입니다. 첫번째 인수가 2진 문자열일 경우, 두 번째 인수도 최대 길이가 1 048 576 바이트인 2진 문자열이어야 합니다. 세번째 인수는 INTEGER 또는 SMALLINT여야 합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

예:

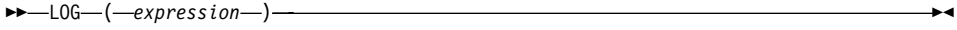
- 단어 'DINING'에서 문자 'N'(처음 발생)의 위치를 찾으십시오.

  ```
  VALUES LOCATE ('N', 'DINING')
  ```

  이 예는 다음 결과를 리턴합니다.

  ```
  1
  -----------
  3
  ```

LOG



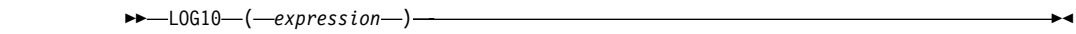
스키마는 SYSFUN입니다.

인수의 자연 로그를 리턴합니다(LN와 동일).

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 &dpfpn;입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

LOG10



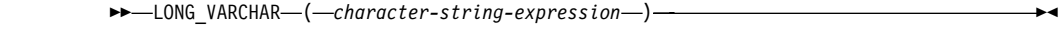
스키마는 SYSFUN입니다.

인수의 기본 10 로그를 리턴합니다.

인수는 내장형 숫자 유형이면 어느 것이나 가능합니다. 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 &dpfpn;입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

LONG_VARCHAR



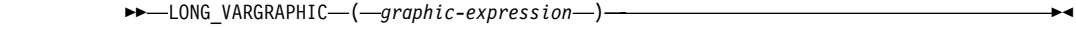
스키마는 SYSIBM입니다.

LONG_VARCHAR 함수는 문자열 데이터 유형의 LONG VARCHAR 표현을 리턴합니다.

문자열 표현식
최대 길이가 32 700인 문자열 값을 리턴하는 *표현식*.

함수의 결과는 LONG VARCHAR입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

LONG_VARGRAPHIC



스키마는 SYSIBM입니다.

LONG_VARGRAPHIC 함수는 2바이트 문자열의 LONG VARGRAPHIC 표현을 리턴합니다.

그래픽 표현식
최대 길이가 16 350 2바이트인 그래픽 문자열 값을 리턴하는 *표현식*.

함수는 결과는 LONG VARGRAPHIC입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

LTRIM

스키마는 SYSIBM입니다.[43]

LTRIM 함수는 *문자열 표현식*에서 공백을 제거합니다.

인수는 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 될 수 있습니다.

- 인수가 DBCS 또는 EUC 데이터베이스에 있는 그래픽 문자열이면, 앞에 있는 2바이트 공백들이 제거됩니다.
- 인수가 유니코드 데이터베이스에 있는 그래픽 문자열이면, 앞에 있는 UCS-2 공백들이 제거됩니다.
- 그렇지 않으면, 앞에 나오는 1바이트 공백들이 제거됩니다.

함수의 결과 데이터 유형은 다음과 같습니다.

- *문자열 표현식*의 데이터 유형이 VARCHAR 또는 CHARVARCHAR인 경우에는 VARCHAR
- *문자열 표현식*의 데이터 유형이 VARGRAPHIC 또는 GRAPHIC인 경우에는 VARGRAPHIC

리턴된 유형의 길이 매개변수는 인수 데이터 유형의 길이 매개변수와 동일합니다.

문자열 표현식에 대한 결과의 실제 길이는 *문자열 표현식*의 길이에서 공백 문자에 대해 제거된 바이트 수를 뺀 것입니다. 그래픽 문자열에 대한 결과의 실제 길이는 *문자열 표현식*의 길이(2바이트 문자 수로 표시)에서 제거된 2바이트 공백 문자 수를 뺀 것입니다. 모든 문자가 제거되면, 결과는 공백의 가변 길이 문자열(길이는 0)입니다.

인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

---

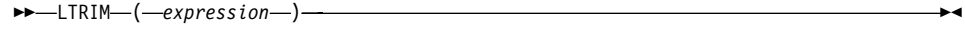
43. 이 함수의 SYSFUN 버전이 LONG VARCHAR 및 CLOB 인수에 대한 지원과 계속 사용 가능합니다. 360 페이지의 『LTRIM(SYSFUN 스키마)』에서 자세한 내용을 참조하십시오.

예: 호스트 변수 HELLO가 CHAR(9)로 정의되며 'Hello'의 값을 가진다고 가정하십시오.

```
VALUES LTRIM(:HELLO)
```

결과는 'Hello'입니다.

LTRIM(SYSFUN 스키마)



스키마는 SYSFUN입니다.

선행 공백이 제거된 상태의 인수의 문자를 리턴합니다.

인수는 내장형 문자열 유형이면 어느 것이나 가능합니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB의 최대 길이는 1 048 576바이트입니다.

함수의 결과는 다음과 같습니다.

- 인수가 VARCHAR(4 000 이하) 또는 CHAR이면 VARCHAR(4000)입니다.
- 인수가 CLOB 또는 LONG VARCHAR이면 CLOB(1M)입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

MICROSECOND

스키마는 SYSIBM입니다.

MICROSECOND 함수는 값의 마이크로초 부분을 리턴합니다.

인수는 시각, 시각 기간 또는 CLOB도 LONG VARCHAR도 아닌 시간의 유효한 문자열 표현이어야 합니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 시간소인 또는 시간소인의 유효한 문자열 표현인 경우:
  - 정수 범위는 0에서 999 999입니다.
- 인수가 기간인 경우:
  - 결과는 정수가 -999 999와 999 999 사이인 값의 마이크로초 부분을 반영합니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- TABLEA 테이블에 유형이 TIMESTAMP인 TS1과 TS2의 두 컬럼이 있다고 가정합니다. TS1의 마이크로초 부분이 0이 아니고 TS1과 TS2의 두 번째 부분이 같은 모든 행을 선택합니다.

```
SELECT * FROM TABLEA
  WHERE MICROSECOND(TS1) <> 0 AND
  SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS

스키마는 SYSFUN입니다.

자정과 인수에서 지정된 시간 값 사이의 시간(초 단위)을 나타내는 0 - 86 400 범위의 정수 값을 리턴합니다.

인수는 시간, 시각 또는 CLOB도 LONG VARCHAR도 아닌 시간이나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

예:

- 밤 12시에서 00:10:10, 밤12시에서 13:10:00사이 초의 수를 찾으십시오.

  ```
  VALUES (MIDNIGHT_SECONDS('00:10:10'), MIDNIGHT_SECONDS('13:10:10'))
  ```

  이 예는 다음 결과를 리턴합니다.

  ```
  1           2
  ----------- -----------
          610       47410
  ```

  1분이 60초이므로, 밤 12시와 지정된 시간 사이에는 610초가 있습니다. 두 번째 예에 대해서도 동일합니다. 1시간이 3600초이고 1분이 60초이므로, 결과 지정된 시간과 밤 12시 사이에는 47410초가 있습니다.

- 밤 12시에서 24:00:00, 밤 12시에서 13:10:00사이 초의 수를 찾으십시오.

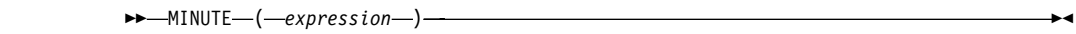
  ```
  VALUES (MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00'))
  ```

  이 예는 다음 결과를 리턴합니다.

  ```
  1           2
  ----------- -----------
        86400           0
  ```

  두 값이 동일한 지점을 나타내지만, 다른 MIDNIGHT_SECONDS 값을 리턴한다는 사실에 유의하십시오.

MINUTE



스키마는 SYSIBM입니다.

MINUTE 함수는 값의 분 부분을 리턴합니다.

인수는 시간, 시각, 시간 기간, 시각 기간 또는 CLOB도 LONG VARCHAR도 아닌 시간이나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

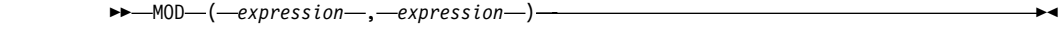
- 인수가 시간, 시간소인 또는 시간이나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 범위가 0 - 59인 값의 분(minute) 부분입니다.
- 인수가 시간 기간이거나 시각 기간인 경우:
  - 결과는 값의 분(minute) 부분으로, 1 - 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- CL_SCHED 샘플 테이블을 사용하여, 지속기간이 50분 이하인 모든 수업을 선택합니다.

```
SELECT * FROM CL_SCHED
  WHERE HOUR(ENDING - STARTING) = 0 AND
  MINUTE(ENDING - STARTING) < 50
```

MOD



스키마는 SYSFUN입니다.

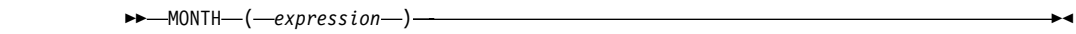
두 번째 인수로 나눈 첫번째 인수의 나머지 부분을 리턴합니다. 첫번째 인수가 음수일 경우에만 결과가 음수입니다.

함수의 결과는 다음과 같습니다.

- 두 인수가 모두 SMALLINT이면 결과는 SMALLINT입니다
- 한 인수는 INTEGER이고 다른 하나는 INTEGER 또는 SMALLINT이면 결과는 INTEGER입니다
- 한 인수는 BIGINT이고 다른 하나는 BIGINT, INTEGER 또는 SMALLINT이면 결과는 BIGINT입니다

결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

MONTH



스키마는 SYSIBM입니다.

MONTH 함수는 값의 월(month) 부분을 리턴합니다.

인수는 날짜, 시각, 날짜 기간, 시각 기간 또는 CLOB도 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 범위가 1 - 12인 값의 월 부분입니다.
- 인수가 날짜 기간이거나 시각 기간인 경우:
  - 결과는 값의 달(month) 부분으로, 1 - 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- DECEMBER에 태어난(BIRTHDATE) 사람에 대한 모든 행을 EMPLOYEE 테이블에서 선택합니다.

```
SELECT * FROM EMPLOYEE
  WHERE MONTH(BIRTHDATE) = 12
```

MONTHNAME

스키마는 SYSFUN입니다.

데이터베이스가 시작될 때의 로케일에 근거하여 인수의 월(month) 부분에 대한 월 이름(예: 1월)이 들어 있는 대소문자 혼합 문자열을 리턴합니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 VARCHAR(100)입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

NODENUMBER

스키마는 SYSIBM입니다.

NODENUMBER 함수는 행의 파티션 번호를 리턴합니다. 예를 들어, SELECT 절에 NODENUMBER가 사용될 경우, SELECT문의 결과를 작성하는 데 사용된 테이블의 각 행에 대한 파티션 번호를 리턴합니다.

전이 변수 및 테이블에서 리턴되는 파티션 번호는 파티션 키 컬럼의 현재 전이 값으로부터 추출됩니다. 예를 들어, 이전(before) 삽입 트리거에서, 새로운 전이 변수의 현재 값을 제공하면 이 함수는 프로젝트된 파티션 번호를 리턴합니다. 그러나, 파티션 키 컬럼 값은 뒤에 나오는 이전(before) 삽입 트리거에 의해 수정될 수 있습니다. 따라서, 데이터베이스에 삽입될 때 행의 마지막 파티션 번호는 프로젝트 값과 다를 수 있습니다.

인수는 테이블의 컬럼에 대한 규정화된 이름이거나, 규정화되지 않은 이름이어야 합니다. 컬럼의 데이터 유형에는 제한이 없습니다. [44] *컬럼 이름*이 뷰(view)의 컬럼을 가리킬 경우, 그 컬럼에 대한 뷰의 표현식은 기준 테이블의 컬럼을 지칭해야 하고 뷰는 삭제 가능해야 합니다. 중첩 또는 공통 테이블 표현식은 뷰와 동일한 규칙을 따릅니다. 삭제 가능한 뷰의 정의에 대해서는 941 페이지의 『주』에서 참조하십시오.

NODENUMBER 함수에서 리턴하는 파티션 번호에 대한 특정 행(및 테이블)은 그 함수를 사용하는 SQL문의 문맥에 의해 결정됩니다.

결과의 데이터 유형은 INTEGER이며 절대 널(NULL)이 아닙니다. 행 레벨 정보가 리턴되므로, 테이블에 지정되는 컬럼에 관계없이 결과는 동일합니다. `db2nodes.cfg` 파일이 없을 경우, 결과는 0입니다.

NODENUMBER 함수는 점검 제한조건 내에서 복제된 테이블에서, 또는 생성된 컬럼의 정의에서 사용할 수 없습니다(SQLSTATE 42881).

---

44. 사용자 정의 함수(UDF) 작성시, 이 함수를 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 모든 데이터 유형을 허용하므로, 사용자 정의 구별 유형을 지원하기 위해 별도의 시그니처를 작성할 필요가 없습니다.

예:

- EMPLOYEE에 대한 행이 DEPARTMENT에 있는 직원의 부서 이름과 다른 파티션상에 있는 행의 수를 셉니다.

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
        WHERE D.DEPTNO=E.WORKDEPT
          AND NODENUMBER(E.LASTNAME) <>NODENUMBER(D.DEPTNO)
```

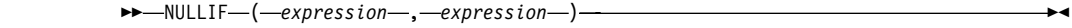
- 두 테이블의 행이 동일한 파티션에 있는 EMPLOYEE와 DEPARTMENT를 조인합니다.

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
        WHERE NODENUMBER(E.LASTNAME) = NODENUMBER(D.DEPTNO)
```

- EMPLOYEE 테이블에서 이전(before) 트리거를 작성함으로써, 사원을 삽입할 경우, EMPINSERTLOG1 테이블에 새로운 행으로 된 사원 번호 및 프로젝트 파티션 번호를 기록합니다.

```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
FOR EACH MODE ROW MODE DB2SQL
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, NODENUMBER(NEWTABLE.EMPNO))
```

NULLIF



스키마는 SYSIBM입니다.

NULLIF 함수는 인수가 같을 경우에는 널(NULL)값을 리턴하고, 그 외에는 첫번째 인수값을 리턴합니다.

인수는 호환 가능해야 합니다(109 페이지의 『지정 및 비교』 참조). 인수는 내장(long 문자열이나 DATALINK가 아님) 또는 구별 데이터 유형(long 문자열이나 DATALINK에 기초한 것이 아님)일 수 있습니다. [45]결과의 속성은 첫번째 인수의 속성입니다.

NULLIF(e1,e2)를 사용한 결과는 다음 표현식을 사용한 결과와 같습니다.

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

e1=e2가 알 수 없음으로 평가되면(하나 또는 두 인수가 NULL이어서), CASE 표현식은 이것을 참이 아닌 것으로 간주합니다. 그러므로, 이 상황에서 NULLIF는 첫번째 인수의 값을 리턴합니다.

예:

- 호스트 변수 PROFIT, CASH, LOSSES의 데이터 유형이 DECIMAL이고, 값이 각각 4500.00, 500.00, 5000.00인 것으로 가정합시다.

  ```
  NULLIF (:PROFIT + :CASH , :LOSSES )
  ```

  널(NULL) 값을 리턴합니다.

45. 사용자 정의 함수(UDF) 작성시, 이 함수를 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 호환성 있는 데이터 유형을 허용하므로, 사용자 정의 구별 유형을 지원하는 추가 시그니처를 작성할 필요가 없습니다.

PARTITION

스키마는 SYSIBM입니다.

PARTITION 함수는 행의 파티션 키 값에 파티션 함수를 적용하여 얻어진 행의 파티션 맵 색인을 리턴합니다. 예를 들어 SELECT절에 사용하는 경우, SELECT 문의 결과를 작성하는 데 사용된 테이블의 각 행에 대한 파티션 맵 색인을 리턴합니다.

전이 변수 및 테이블에서 리턴되는 파티션 맵 색인은 파티션 키 컬럼의 현재 전이 값으로부터 추출됩니다. 예를 들어, 이전(before) 삽입 트리거에서 새로운 전이 변수의 현재 값을 제공하면 이 함수는 프로젝트된 파티션 맵 색인을 리턴합니다. 그러나, 파티션 키 컬럼 값은 뒤에 나오는 이전(before) 삽입 트리거에 의해 수정될 수 있습니다. 따라서, 데이터베이스에 삽입될 때 행의 최종 파티션 맵 색인은 프로젝트 값과 다를 수 있습니다.

인수는 테이블의 컬럼에 대한 규정화된 이름이거나, 규정화되지 않은 이름이어야 합니다. 컬럼의 데이터 유형에는 제한이 없습니다. [46] *컬럼 이름*이 뷰(view)의 컬럼을 가리킬 경우, 그 컬럼에 대한 뷰의 표현식은 기준 테이블의 컬럼을 지칭해야 하고 뷰는 삭제 가능해야 합니다. 중첩 또는 공통 테이블 표현식은 뷰와 동일한 규칙을 따릅니다. 삭제 가능한 뷰의 정의에 대해서는 941 페이지의 『주』에서 참조하십시오.

PARTITION 함수에서 리턴하는 파티션 맵 색인에 대한 특정 행(및 테이블)은 그 함수를 사용하는 SQL문의 문맥에 의해 결정됩니다.

결과의 데이터 유형은 0 - 4095 범위의 INTEGER입니다. 파티션 키가 없는 테이블의 경우, 결과는 항상 0입니다. 널(NULL) 값은 리턴되지 않습니다. 행 레벨 정보가 리턴되므로, 테이블에 지정되는 컬럼에 관계없이 결과는 동일합니다.

---

46. 사용자 정의 함수(UDF) 작성시, 이 함수를 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 모든 데이터 유형을 허용하므로, 사용자 정의 구별 유형을 지원하기 위해 별도의 시그니처를 작성할 필요가 없습니다.

PARTITION 함수는 점검 제한조건 내에서 복제된 테이블에서, 또는 생성된 컬럼의 정의에서 사용할 수 없습니다(SQLSTATE 42881).

예:

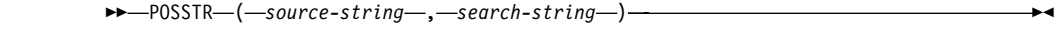
- 파티션 맵 색인 100을 갖는 모든 행에 대해 EMPLOYEE 테이블의 직원 수(EMPNO)를 나열합니다.

```
SELECT EMPNO FROM EMPLOYEE
  WHERE PARTITION(PHONENO) =  100
```

- EMPLOYEE 테이블에서 이전(before) 트리거를 작성함으로써, 사원을 삽입할 경우 EMPINSERTLOG2 테이블에 새로운 행으로 된 프로젝트 파티션 맵 색인 및 사원 번호를 기록합니다.

```
CREATE TRIGGER EMPINSLOGTRIG2
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
FOR EACH MODE ROW MODE DB2SQL
INSERT INTO EMPINSERTLOG2
  VALUES(NEWTABLE.EMPNO, PARTITION(NEWTABLE.EMPNO))
```

POSSTR



스키마는 SYSIBM입니다.

POSSTR 함수는 한 문자열(*소스-문자열*) 내에서 다른 문자열(*검색 문자열*)이 처음 나타난 시작 위치를 리턴합니다. *검색 문자열* 위치에 대한 숫자는 1(0이 아님)에서 시작합니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

소스 문자열
검색이 수행될 소스 문자열을 지정하는 표현식.

표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 호스트 변수(위치 지정자 변수나 파일 참조 변수를 포함하여)
- 스칼라 함수
- 대형 오브젝트(LOB) 위치 지정자
- 컬럼 이름
- 위의 설명을 병합한 표현식

검색 문자열
검색될 문자열을 지정하는 표현식.

표현식은 다음 중 하나로 지정할 수 있습니다.

- 상수
- 특수 레지스터
- 호스트 변수
- 피연산자가 위의 사항 중 하나인 스칼라 함수
- 위의 설명을 병합한 표현식

다음과 같은 제한을 받습니다.

- 표현식의 어떤 요소도 LONG VARCHAR, CLOB, LONG VARGRAPHIC 또는 DBCLOB이 될 수 없습니다. 또한, BLOB 파일 참조 변수가 될 수 없습니다.
- *검색-문자열*의 실제 길이는 4 000바이트보다 클 수 없습니다.

이러한 규칙들은 227 페이지의 『LIKE 술어』에서 설명된 *패턴 표현식*에 대한 규칙과 같습니다.

*검색-문자열*과 *소스-문자열* 둘다는 0 또는 그 이상의 연속 위치를 갖습니다. 문자열이 문자이거나 2진 문자열이면, 위치는 1바이트입니다. 문자열이 그래픽 문자열이면, 위치도 그래픽(DBCS) 문자입니다.

POSSTR 함수는 혼합 데이터열을 받아들입니다. 그러나 POSSTR은 1바이트와 복수 바이트 문자간의 변경에 관계없이, 엄격한 바이트 계산 기준으로 작동됩니다.

다음 규칙이 적용됩니다.

- *소스-문자열*과 *검색-문자열*의 데이터 유형은 호환성이 있어야 하며, 그렇지 않은 경우에는 오류가 발생합니다(SQLSTATE 42884).
  - *소스-문자열*이 문자열이면, *검색-문자열*은 CLOB 또는 LONG VARCHAR이 아닌 문자열이어야 하며, 그 실제 길이는 32 672바이트 이하여야 합니다.
  - *소스-문자열*이 그래픽 문자열이면, *검색-문자열*은 DBCLOB 또는 LONG VARGRAPHIC이 아닌 그래픽 문자열이어야 하며, 그 실제 길이는 16 336 2바이트 문자 이하여야 합니다.
  - *소스-문자열*이 2진 문자열이면, *검색-문자열*은 실제 길이가 32 672바이트 이하인 2진 문자열이어야 합니다.
- *검색* 문자열의 길이가 0이면, 함수에서 리턴되는 결과는 1입니다.
- 그렇지 않을 경우:
  - *소스-문자열*의 길이가 0이면, 함수에서 리턴되는 길이는 0입니다.
  - 그렇지 않을 경우:
    - *검색-문자열*의 값이 *소스-문자열*의 값에서 연속 위치의 동일 길이의 부속 문자열과 같으면, 함수에 의해 리턴되는 결과는 *소스-문자열* 값 내의 그런 부속 문자열의 첫번째의 시작 위치입니다.

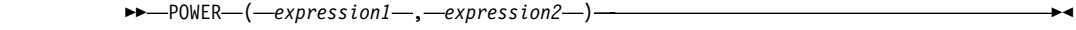
- 그렇지 않으면, 함수에 의해 리턴되는 결과는 0입니다.

예

- 'GOOD BEER' 단어를 포함하는 IN_TRAY 테이블에 있는 모든 항목에 대해 NOTE_TEXT 컬럼 내에서 그 단어들의 시작 위치와 RECEIVED 및 SUBJECT 컬럼을 선택합니다.

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD BEER')
  FROM IN_TRAY
  WHERE POSSTR(NOTE_TEXT, 'GOOD BEER') <> 0
```

POWER



스키마는 SYSFUN입니다.

*표현식1*의 값을 *표현식2*의 승수로 리턴합니다.

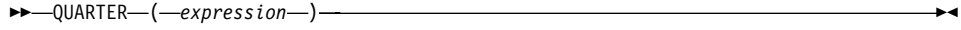
인수는 내장형 데이터 유형이면 어느 것이나 될 수 있습니다. DECIMAL 및 REAL 인수는 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 다음과 같습니다.

- 인수가 INTEGER 또는 SMALLINT이면 INTEGER입니다.
- 한 인수는 BIGINT이고 다른 하나는 BIGINT, INTEGER 또는 SMALLINT이면 BIGINT입니다
- 그렇지 않으면, DOUBLE입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

QUARTER



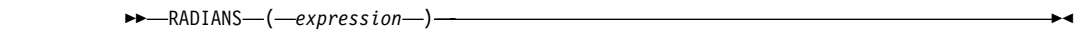
스키마는 SYSFUN입니다.

인수에 지정된 날짜에 대한 분기를 나타내는 1-4 범위의 정수 값을 리턴합니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

RADIANS



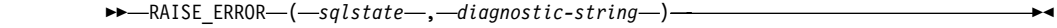
스키마는 SYSFUN입니다.

도(degree)로 표시된 인수에서 변환된 라디안의 수를 리턴합니다.

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 &dpfpn;입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

RAISE_ERROR



스키마는 SYSIBM입니다.

RAISE_ERROR 함수는 이 함수를 포함하는 명령문이 지정된 SQLSTATE, SQLCODE -438 및 *진단 문자열*과 함께 오류를 리턴하도록 합니다. RAISE_ERROR 함수는 항상 정의되지 않은 데이터 유형에 대해 NULL을 리턴합니다.

sqlstate

정확히 5자를 포함하는 문자열입니다. 이것은 길이 5로 정의된 CHAR 유형이거나 길이 5 이상으로 정의된 VARCHAR 유형이어야 합니다. *sqlstate* 값은 다음과 같이 응용프로그램 정의 SQLSTATE에 대한 규칙을 따라야 합니다.

- 각 문자는 숫자 집합('0' - '9')의 문자이거나, 강조되지 않은 대문자('A' - 'Z')여야 합니다.
- SQLSTATE 클래스(처음 두 자)는 오류 클래스가 아니므로 '00', '01' 또는 '02'가 될 수 없습니다.
- SQLSTATE 클래스(처음 두 자)가 문자 '0' - '6'이나 'A' - 'H'로 시작할 경우, 부속클래스(마지막 세 자)는 범위 'I' - 'Z' 범위 내의 문자로 시작해야 합니다.
- SQLSTATE 클래스(처음 두 자)가 문자 '7', '8', '9' 또는 'I' - 'Z'로 시작할 경우, 부속클래스(마지막 세 자)는 '0' - '9' 또는 'A' - 'Z'가 될 수 있습니다.

SQLSTATE가 이 규칙에 따르지 않으면 오류가 발생합니다(SQLSTATE 428B3).

진단 문자열

오류 상태를 설명하는 최대 70바이트까지의 문자열을 리턴하는 CHAR 또는 VARCHAR 유형의 표현식입니다. 문자열이 70바이트보다 크면, 그 문자열은 절단됩니다.

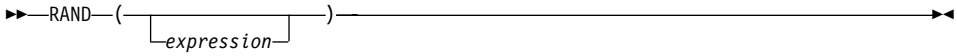
결과 데이터 유형에 대한 규칙이 적용되지 않는 문맥에서 이 함수를 사용하려면, 데이터 유형에 널(NULL) 값이 리턴되도록 cast 스펙을 사용해야 합니다. CASE 표현식은 RAISE_ERROR 함수가 가장 유용한 표현식입니다.

예:

사원 번호와 교육 레벨을 Post Graduate, Graduate 및 Diploma로 나열합니다. 교육 레벨이 20보다 크면, 오류가 발생합니다.

```
SELECT EMPNO,
       CASE WHEN EDUCLVL < 16 THEN 'Diploma'
            WHEN EDUCLVL < 18 THEN 'Graduate'
            WHEN EDUCLVL < 21 THEN 'Post Graduate'
            ELSE RAISE_ERROR('70001',
                             'EDUCLVL has a value greater than 20')
      END
                        FROM EMPLOYEE
```

RAND



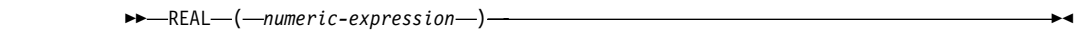
스키마는 SYSFUN입니다.

인수를 선택적 시드(seed) 값으로 사용하여 0에서 1 사이의 임의의 부동 소수점 값을 리턴합니다. 이 함수는 결정되지 않는 함수로 정의됩니다.

인수가 필요하지는 않지만, 지정될 경우 INTEGER 또는 SMALLINT가 될 수 있습니다.

함수의 결과는 &dpfpn;입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

REAL



스키마는 SYSIBM입니다.

REAL 함수는 숫자의 단일 정밀도 부동 소수점 표현을 리턴합니다.

인수는 내장 숫자 데이터 유형값을 리턴하는 표현식입니다.

함수의 결과는 단일 정밀도 부동 소수점 수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

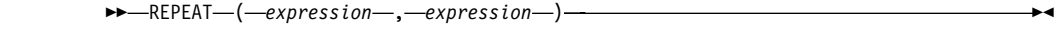
결과는 인수가 단일 정밀도 부동 소수점 컬럼이나 변수에 지정되는 경우에 발생하는 것과 같은 수입니다.

예:

EMPLOYEE 테이블을 사용하여, 수당이 0이 아닌 사원들의 수당에 대한 급여 비율을 찾습니다. 관련된 컬럼(SALARY 및 COMM)은 DECIMAL 데이터 유형을 갖습니다. 결과는 단일 정밀도 부동 소수점이어야 합니다. 따라서, 부동 소수점(실질적으로 배정밀도)에서 나눗셈이 수행되도록 SALARY에 REAL이 적용된 다음 단일 정밀도의 부동 소수점을 리턴하도록 REAL이 완전한 표현식에 적용됩니다.

```
SELECT EMPNO, REAL(REAL(SALARY)/COMM)
                          FROM EMPLOYEE
  WHERE COMM > 0
```

REPEAT



스키마는 SYSFUN입니다.

두 번째 인수가 지정하는 배수만큼 반복되는 첫번째 인수로 구성되는 문자열을 리턴합니다.

첫번째 인수는 문자열 또는 2진 문자열 유형입니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB 또는 2진 문자열의 최대 길이는 1 048 576바이트입니다. 두 번째 인수는 SMALLINT 또는 INTEGER가 될 수 있습니다.

함수의 결과는 다음과 같습니다.

- 첫번째 인수가 VARCHAR(4 000 이하) 또는 CHAR이면 VARCHAR(4000)입니다.
- 첫번째 인수가 CLOB 또는 LONG VARCHAR이면 CLOB(1M)입니다.
- 첫번째 인수가 BLOB이면 BLOB(1M)입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

예:

- 구 'REPEAT THIS'를 5번 나열하십시오.

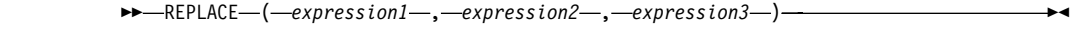
  ```
  VALUES CHAR(REPEAT('REPEAT THIS', 5), 60)
  ```

  이 예는 다음을 리턴합니다.

  ```
  1
  ------------------------------------------------------------
  REPEAT THISREPEAT THISREPEAT THISREPEAT THISREPEAT THIS
  ```

  언급한 대로, REPEAT 함수의 출력은 VARCHAR(4000)입니다. 위의 예에서 함수 CHAR는 REPEAT의 출력을 60바이트로 제한하는 데 사용됩니다.

REPLACE



스키마는 SYSFUN입니다.

*표현식1*에 있는 모든 *표현식2*를 *표현식3*으로 대체합니다.

첫번째 인수는 내장형 문자열 또는 2진 문자열 유형이면 어느 것이나 가능합니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB 또는 2진 문자열의 최대 길이는 1 048 576바이트입니다. CHAR은 VARCHAR로 변환되고 LONG VARCHAR는 CLOB(1M)로 변환됩니다. 두번째와 세번째 인수는 첫번째 인수와 동일합니다.

함수의 결과는 다음과 같습니다.

- 첫번째, 두번째, 세번째 인수가 VARCHAR 또는 CHAR이면 VARCHAR(4000)입니다.
- 첫번째, 두번째, 세번째 인수가 CLOB 또는 LONG VARCHAR이면 CLOB(1M)입니다.
- 첫번째, 두번째, 세번째 인수가 BLOB이면 BLOB(1M)입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

예:

- 단어 'DINING'에서 모든 문자 'N'을 'VIP'로 대체하십시오.

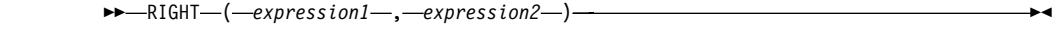
  ```
  VALUES CHAR (REPLACE ('DINING', 'N', 'VID'), 10)
  ```

  이 예는 다음 결과를 리턴합니다.

  ```
  1
  ----------
  DIVIDIVIDG
  ```

  언급한 대로, REPLACE 함수의 출력은 VARCHAR(4000)입니다. 위의 예에서 함수 CHAR은 REPLACE의 출력을 10바이트로 제한하는 데 사용됩니다.

RIGHT



스키마는 SYSFUN입니다.

*표현식1*의 가장 왼쪽 *표현식2* 바이트로 구성되는 문자열을 리턴합니다. *표현식1* 값은 *표현식1*의 지정된 부속 문자열이 항상 존재하도록 효율적으로 필요한 만큼의 공백으로 오른쪽이 채워집니다.

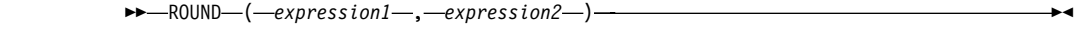
첫번째 인수는 문자열 또는 2진 문자열 유형입니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB 또는 2진 문자열의 최대 길이는 1 048 576바이트입니다. 두번째 인수는 INTEGER 또는 SMALLINT가 될 수 있습니다.

함수의 결과는 다음과 같습니다.

- 첫번째 인수가 VARCHAR(4 000 이하) 또는 CHAR이면 VARCHAR(4000)입니다.
- 첫번째 인수가 CLOB 또는 LONG VARCHAR이면 CLOB(1M)입니다.
- 첫번째 인수가 BLOB이면 BLOB(1M)입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

ROUND



스키마는 SYSFUN입니다.

소수점의 오른쪽 *표현식2* 자리로 반올림된 *표현식1*을 리턴합니다. *표현식2*가 음수이면, *표현식1*은 소수점 왼쪽에 있는 *표현식2* 자리의 절대값으로 반올림됩니다.

첫번째 인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 두 번째 인수는 INTEGER 또는 SMALLINT가 될 수 있습니다. 함수가 처리할 수 있도록 DECIMAL과 REAL은 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 다음과 같습니다.

- 첫번째 인수가 INTEGER 또는 SMALLINT일 경우 INTEGER입니다.
- 첫번째 인수가 BIGINT일 경우 BIGINT입니다
- 첫번째 인수가 DOUBLE, DECIMAL 또는 REAL일 경우 DOUBLE입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

예:

- 2, 1, 0, -1 및 -2 소수 자리로 각각 반올림된 수 873.726을 표시하십시오.

```
VALUES (DECIMAL(ROUND(873.726,2),6,3), DECIMAL(ROUND(873.726,1),6,3),
        DECIMAL(ROUND(873.726,0),6,3), DECIMAL(ROUND(873.726,-1),6,3),
        DECIMAL(ROUND(873.726,-2),6,3))
```

  위의 예는 다음을 리턴합니다.

```
1        2        3        4        5
-------- -------- -------- -------- ---------
 873.730  873.700  874.000  870.000   900.000
```

  언급한 대로, ROUND 함수의 출력은 DOUBLE입니다. 위의 예에서 함수 DECIMAL는 ROUND의 출력을 제한하는 데 사용됩니다.

RTRIM

스키마는 SYSIBM입니다.[47]

RTRIM 함수는 *문자열 표현식*의 끝에서 공백을 제거합니다.

인수는 CHAR, VARCHAR, GRAPHIC 또는 VARGRAPHIC 데이터 유형이 될 수 있습니다.

- 인수가 DBCS 또는 EUC 데이터베이스에 있는 그래픽 문자열이면, 뒤에 있는 2바이트 공백들이 제거됩니다.
- 인수가 유니코드 데이터베이스에 있는 그래픽 문자열이면, 뒤에 있는 UCS-2 공백들이 제거됩니다.
- 그렇지 않으면, 뒤에 오는 1바이트 공백들이 제거됩니다.

함수의 결과 데이터 유형은 다음과 같습니다.

- *문자열 표현식*의 데이터 유형이 VARCHAR 또는 CHARVARCHAR인 경우에는 VARCHAR
- *문자열 표현식*의 데이터 유형이 VARGRAPHIC 또는 GRAPHIC인 경우에는 VARGRAPHIC

리턴된 유형의 길이 매개변수는 인수 데이터 유형의 길이 매개변수와 동일합니다.

문자열 표현식에 대한 결과의 실제 길이는 *문자열 표현식*의 길이에서 공백 문자에 대해 제거된 바이트 수를 뺀 것입니다. 그래픽 문자열에 대한 결과의 실제 길이는 *문자열 표현식*의 길이(2바이트 문자 수로 표시)에서 제거된 2바이트 공백 문자 수를 뺀 것입니다. 모든 문자가 제거되면, 결과는 공백의 가변 길이 문자열(길이는 0)입니다.

인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

---

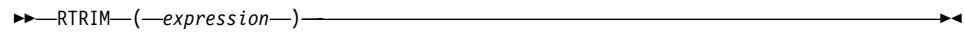
47. 이 함수의 SYSFUN 버전이 LONG VARCHAR 및 CLOB 인수에 대한 지원과 계속 사용 가능합니다. 388 페이지의 『RTRIM(SYSFUN 스키마)』에서 자세한 내용을 참조하십시오.

예: 호스트 변수 HELLO가 CHAR(9)로 정의되며 'Hello'의 값을 가진다고 가정하십시오.

```
VALUES RTRIM(:HELLO)
```

결과는 'Hello'입니다.

RTRIM(SYSFUN 스키마)



스키마는 SYSFUN입니다.

뒷 공백이 제거된 상태의 인수의 문자를 리턴합니다.

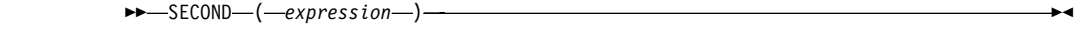
인수는 내장형 문자열 데이터 유형이면 어느 것이나 가능합니다. VARCHAR의 최대 길이는 4 000바이트이고 CLOB의 최대 길이는 1 048 576바이트입니다.

함수의 결과는 다음과 같습니다.

- 인수가 VARCHAR(4 000 이하) 또는 CHAR이면 VARCHAR(4000)입니다.
- 인수가 CLOB 또는 LONG VARCHAR이면 CLOB(1M)입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

SECOND



스키마는 SYSIBM입니다.

SECOND 함수는 값의 초 부분을 리턴합니다.

인수는 시간, 시각, 시간 기간, 시각 기간 또는 CLOB도 LONG VARCHAR도 아닌 시간이나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 시간, 시간소인 또는 시간이나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 범위 0 - 59 사이의 정수인 값의 초(second) 부분입니다.
- 인수가 시간 기간이거나 시각 기간인 경우:
  - 결과는 값의 초(second) 부분으로, 1 - 99 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- 호스트 변수 TIME_DUR (decimal(6,0))의 값이 153045인 것으로 가정합니다.

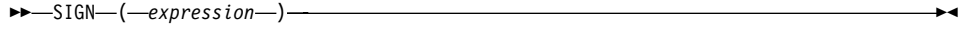
  ```
  SECOND(:TIME_DUR)
  ```

  값 45를 리턴합니다.
- RECEIVED 컬럼(시각)에 '1988-12-25-17.12.30.000000'에 해당되는 내부 값을 갖고 있는 것으로 가정합니다.

  ```
  SECOND(RECEIVED)
  ```

  값 30을 리턴합니다.

SIGN



스키마는 SYSFUN입니다.

인수의 부호 표시기(indicator)를 리턴합니다. 인수가 0미만일 경우, -1이 리턴됩니다. 인수가 0과 같으면 0이 리턴됩니다. 인수가 0보다 크면, 1이 리턴됩니다.

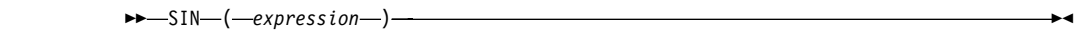
인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 함수가 처리할 수 있도록 DECIMAL과 REAL은 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 다음과 같습니다.

- 인수가 SMALLINT이면 SMALLINT입니다.
- 인수가 INTEGER이면 INTEGER입니다.
- 인수가 BIGINT이면 BIGINT입니다.
- 그렇지 않으면, DOUBLE입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

SIN



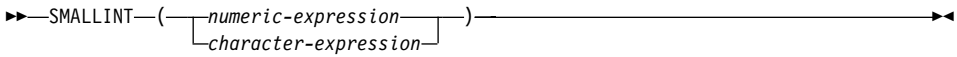
스키마는 SYSFUN입니다.

인수의 사인(sine)을 리턴하며, 여기서 인수는 라디안으로 표시되는 각도입니다.

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

SMALLINT



스키마는 SYSIBM입니다.

SMALLINT 함수는 작은 정수 상수의 형식으로 숫자의 정수 표현이나 문자열을 리턴합니다.

숫자 표현식

임의의 내장 숫자 데이터 유형값을 리턴시키는 표현식.

인수가 *숫자-표현식*이면, 결과는 인수가 작은 정수 컬럼 또는 변수에 지정되었을 경우에 발생한 것과 같은 수입니다. 인수의 모든 부분이 작은 정수 범위 내에 있지 않으면, 오류가 발생합니다. 인수의 소수 부분이 있으면, 절단됩니다.

문자 표현식

문자 상수의 최대 길이보다 작은 길이를 갖는 문자열 값을 리턴하는 표현식입니다. 앞뒤 공백은 제거되고 결과 문자열은 SQL 정수 상수를 형성하는 규칙에 따라야 합니다(SQLSTATE 22018). 그러나, 상수 값은 작은 정수(SQLSTATE 22003) 범위 내에 있어야 합니다. 문자열은 long 문자열이 될 수 없습니다.

인수가 *문자 표현식*이면, 결과는 해당 정수 상수가 작은 정수 컬럼이나 변수에 지정되었을 때 발생한 것과 같은 수입니다.

함수의 결과는 작은 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

SOUNDEX

스키마는 SYSFUN입니다.

인수에서 단어들의 사운드를 나타내는 4자 코드를 리턴합니다. 그 결과는 다른 문자열의 사운드와 비교하는 데 사용됩니다.

인수는 4 000 바이트를 초과하지 않은 CHAR 또는 VARCHAR 중 하나의 문자열이 될 수 있습니다.

함수의 결과는 CHAR(4)입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

SOUNDEX 함수는 발음은 알지만 정확한 철자를 모르는 문자열을 찾는 데 유용합니다. 발음이 유사한 단어를 검색하는 데 도움이 되는 문자와 문자 조합의 발음 방식에 대한 가정을 설정합니다. 문자열을 인수로 하여 DIFFERENCE 함수에 전달하거나 직접 비교를 수행할 수 있습니다(318 페이지의 『DIFFERENCE』 참조).

예:

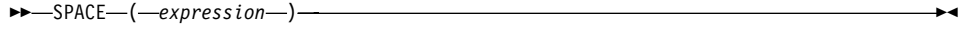
EMPLOYEE 테이블을 이용하여 'Loucesy'와 유사하게 발음되는 성을 가진 직원의 EMPNO와 LASTNAME을 찾습니다.

```
SELECT EMPNO, LASTNAME FROM EMPLOYEE
 WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

이 예는 다음 결과를 리턴합니다.

```
EMPNO  LASTNAME
------ ---------------
000110 LUCCHESSI
```

SPACE



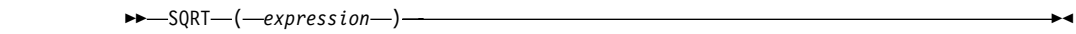
스키마는 SYSFUN입니다.

두 번째 인수가 지정하는 길이의 공백으로 구성되는 문자열을 리턴합니다.

인수는 SMALLINT 또는 INTEGER입니다.

함수의 결과는 VARCHAR(4000)입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

SQRT



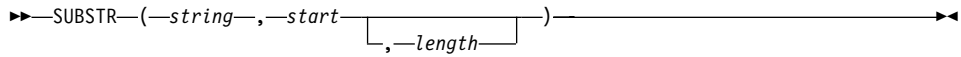
스키마는 SYSFUN입니다.

인수의 제곱근을 리턴합니다.

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환해야 합니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

SUBSTR



스키마는 SYSIBM입니다.

SUBSTR 함수는 문자열의 부속문자열을 리턴합니다.

*문자열*이 문자(character)로 된 문자열이면, 함수의 결과는 첫번째 인수의 코드 페이지로 표시되는 문자열입니다. 이것이 2진 문자열이면, 함수의 결과는 2진 문자열입니다. 이것이 그래픽 문자열이면, 함수의 결과는 첫번째 인수의 코드 페이지로 표시되는 그래픽 문자열입니다. SUBSTR 함수의 인수 중 하나가 널(NULL)이 될 수 있으면, 결과는 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면, 결과는 널(NULL) 값입니다.

문자열

결과가 파생되는 문자열을 지정하는 표현식.

*문자열*이 문자(character)로 된 문자열이거나 2진 문자열이면, *문자열*의 부속문자열은 *문자열*의 0개 이상의 연속 바이트여야 합니다. *문자열*이 그래픽 문자열인 경우, *문자열*의 부속문자열은 *문자열*의 0 또는 그 이상의 연속된 2바이트 문자입니다.

시작

문자로 된 문자열이나 2진 문자열에 대한 결과의 첫번째 바이트 위치 또는 그래픽 문자열에 대한 결과의 첫번째 문자 위치를 지정하는 표현식. *시작*은 *문자열*이 고정 길이인지 가변 길이인지(범위를 벗어날 경우, SQLSTATE 22011)에 따라, 1과 *문자열*의 길이 또는 최대 길이 사이의 정수여야 합니다. 이것은 응용프로그램 코드 페이지가 아니라, 데이터베이스 코드 페이지의 구문에서 바이트 수로 지정해야 합니다.

길이

결과의 길이를 지정하는 표현식. 지정할 경우, *길이*는 0 - *n* 범위 내의 2진 정수여야 합니다. 여기서 *n*은 (*문자열*의 길이 속성) - *start* + 1과 같습니다(범위를 벗어날 경우 SQLSTATE 22011).

*길이*를 명시적으로 지정할 경우, *문자열*은 *문자열*의 지정된 부속 문자열이 항상 존재하도록 효율적으로 필요한 만큼의 공백(문자열의 경우 1 바이트, 그래픽 문자열의 경우 2 바이트)으로 오른쪽이 채워집니다. *길이*의 기본값은 문자로 된 문자열이나 2진 문자열의 경우에, *시작*에 지정된 바이트에서 *문자열*의 마지막 바이트까지의 바이트 수이고, 그래픽 문자열의 경우엔 *시작*에 지정된 문자에서 *문자열*의 마지막 문자까지의 2바이트 문자 수입니다. 그러나, *문자열*이 *시작* 이하의 길이를 갖는 가변 길이 문자열인 경우, 기본값은 0이고, 결과는 빈 문자열입니다(이것은 응용프로그램 코드 페이지가 아니라, 데이터베이스 코드 페이지의 문맥에서 바이트 수로 지정해야 합니다. 예를 들어, 데이터 유형이 VARCHAR(18)이고 값이 'MCKNIGHT'인 NAME 컬럼이 SUBSTR(NAME,10)의 빈 문자열을 생성합니다.).

표16은 입력 유형과 속성에 따라 SUBSTR 함수의 결과 유형과 길이를 보여줍니다.

표 16. SUBSTR 결과의 데이터 유형과 길이

문자열 인수 데이터 유형	길이 인수	결과 데이터 유형
CHAR(A)	상수(*l*<255)	CHAR(*l*)
CHAR(A)	지정되지 않았으나 *시작(start)* 인수는 상수임	CHAR(A-*start*+1)
CHAR(A)	상수가 아님	VARCHAR(A)
VARCHAR(A)	상수(*l*<255)	CHAR(*l*)
VARCHAR(A)	상수(254<*l*<32673)	VARCHAR(*l*)
VARCHAR(A)	상수가 아니거나 지정되지 않음	VARCHAR(A)
LONG VARCHAR	상수(*l*<255)	CHAR(*l*)
LONG VARCHAR	상수(254<*l*<4001)	VARCHAR(*l*)
LONG VARCHAR	상수(*l*>4000)	LONG VARCHAR
LONG VARCHAR	상수가 아니거나 지정되지 않음	LONG VARCHAR
CLOB(A)	상수(*l*)	CLOB(*l*)
CLOB(A)	상수가 아니거나 지정되지 않음	CLOB(A)

표 16. SUBSTR 결과의 데이터 유형과 길이 (계속)

문자열 인수 데이터 유형	길이 인수	결과 데이터 유형
GRAPHIC(A)	상수(*l*<128)	GRAPHIC(*l*)
GRAPHIC(A)	지정되지 않았으나 *시작(start)* 인수는 상수임	GRAPHIC(A-*start*+1)
GRAPHIC(A)	상수가 아님	VARGRAPHIC(A)
VARGRAPHIC(A)	상수(*l*<128)	GRAPHIC(*l*)
VARGRAPHIC(A)	상수(127<*l*<16337)	VARGRAPHIC(*l*)
VARGRAPHIC(A)	상수가 아님	VARGRAPHIC(A)
LONG VARGRAPHIC	상수(*l*<128)	GRAPHIC(*l*)
LONG VARGRAPHIC	상수(127<*l*<2001)	VARGRAPHIC(*l*)
LONG VARGRAPHIC	상수(*l*>2000)	LONG VARGRAPHIC
LONG VARGRAPHIC	상수가 아니거나 지정되지 않음	LONG VARGRAPHIC
DBCLOB(A)	상수(*l*)	DBCLOB(*l*)
DBCLOB(A)	상수가 아니거나 지정되지 않음	DBCLOB(A)
BLOB(A)	상수(*l*)	BLOB(*l*)
BLOB(A)	상수가 아니거나 지정되지 않음	BLOB(A)

*문자열*이 고정 길이 문자열일 경우, *길이*를 생략하면 내재적으로 LENGTH(*string*) - *start* + 1이 지정됩니다. *문자열*이 가변 길이 문자열이면, *길이*를 생략할 경우 0이나 LENGTH(*string*) - *start* + 1 중에서 큰 값이 내재적으로 지정됩니다.

예:

- 호스트 변수 NAME(VARCHAR(50))이 'BLUE JAY' 값을 갖고 호스트 변수 SURNAME_POS(int)가 6 값을 갖는다고 가정합니다.

  ```
  SUBSTR(:NAME, :SURNAME_POS):ehp2s
  ```

  값 'JAY'를 리턴합니다.

  ```
  SUBSTR(:NAME, :SURNAME_POS,1)
  ```

값 'J'를 리턴합니다.

- 프로젝트 이름(PROJNAME)이 단어 'OPERATION '으로 시작하는 모든 행을 PROJECT 테이블에서 선택합니다.

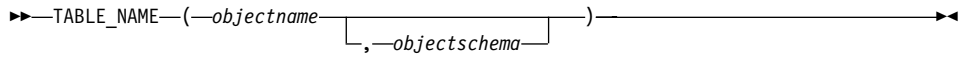
```
SELECT * FROM PROJECT
  WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
```

상수의 끝에 있는 공백은 'OPERATIONS'와 같이 초기 단어를 사용할 때 필요합니다.

주:

1. 동적 SQL에서, *문자열*, *시작* 및 *길이*는 매개변수 표시문자(?)로 나타낼 수 있습니다. 매개변수 표시문자가 *문자열*에 사용된 경우, 피연산자의 데이터 유형은 VARCHAR이고, 그 피연산자는 널(NULL) 입력 가능하게 됩니다.
2. 위의 결과 정의에서 명시적으로 언급하지 않았더라도, 이것은 *문자열*이 1바이트 및 복수 바이트의 혼합일 경우, 결과에는 *시작* 및 *길이*값에 따라 복수 바이트 문자의 단편(fragment)이 들어 있을 수 있다는 의미입니다. 즉, 결과는 2바이트 문자의 두 번째 바이트로 시작하거나, 2바이트 문자의 첫번째 바이트로 끝날 수 있습니다. SUBSTR 함수는 그러한 단편을 감지하지 않고 수행해야 하는 어떤 특수 처리도 제공하지 않습니다.

TABLE_NAME



스키마는 SYSIBM입니다.

TABLE_NAME 함수는 별명 체인이 해석된 이후에 발견된 오브젝트의 규정화되지 않은 이름을 리턴합니다. 지정된 *오브젝트 이름*(그리고 *오브젝트-스키마*)은 해석의 시작점으로 사용됩니다. 시작점이 별명을 참조하지 않은 경우, 규정화된 시작점 이름이 리턴됩니다. 결과로 나오는 이름은 테이블, 뷰 또는 정의되지 않은 오브젝트가 될 수 있습니다.

오브젝트 이름
: 해석될 비규정화된 이름(보통 기존 별명의 이름)을 나타내는 문자 표현식. *오브젝트 이름*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0보다 크고 129자 미만이여야 합니다.

오브젝트-스키마
: 해석에 앞서 공급된 *objectname* 값을 규정화하는 데 사용된 스키마를 나타내는 문자 표현식. *오브젝트 스키마*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0보다 크고 129자 미만이여야 합니다.

  *오브젝트-스키마*가 제공되지 않으면, 규정자에 기본 스키마가 사용됩니다.

함수 결과의 데이터 유형은 VARCHAR(128)입니다. *오브젝트 이름*이 널(NULL)이 될 수 있을 경우, 결과도 널(NULL)이 될 수 있습니다. 즉, *오브젝트 이름*이 널(NULL)이면 결과도 널(NULL) 값입니다. *오브젝트-스키마*가 널(NULL) 값인 경우, 기본 스키마 이름이 사용됩니다. 결과는 규정화되지 않은 이름을 표시하는 문자열입니다. 결과 이름은 다음 중 하나로 표시할 수 있습니다.

테이블 *오브젝트 이름*의 값이, 테이블 이름(입력값이 리턴됨)이거나 해당 이름이 리턴되는 테이블에 대해 해석된 별명이었습니다.

뷰 *오브젝트 이름*의 값이, 뷰 이름(입력값이 리턴됨)이거나 해당 이름이 리턴되는 뷰에 대해 해석된 별명이었습니다.

정의되지 않은 오브젝트

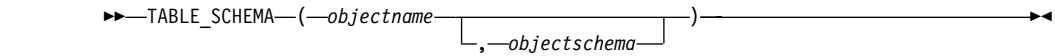
*오브젝트 이름*의 값이, 오브젝트 이름(입력값이 리턴됨)이거나 해당 이름이 리턴되는 정의되지 않은 오브젝트에 대해 해석된 정의되지 않은 별명이었습니다.

그러므로, 이 함수에 널(NULL)이 아닌 값이 제공되면, 결과 이름을 갖고 있는 오브젝트가 전혀 없을 경우에도 값이 항상 리턴됩니다.

예:

402 페이지의 『TABLE_SCHEMA』에 있는 예 부분을 참조하십시오.

TABLE_SCHEMA



스키마는 SYSIBM입니다.

TABLE_SCHEMA 함수는 별명 체인이 해석된 이후에 발견된 오브젝트의 스키마 이름을 리턴합니다. 지정된 *오브젝트 이름*(그리고 *오브젝트-스키마*)은 해석의 시작점으로 사용됩니다. 문자열 시작점이 별명을 참조하지 않은 경우, 시작점의 스키마 이름이 리턴됩니다. 결과로 만들어진 스키마 이름은 테이블, 뷰 또는 정의되지 않은 오브젝트가 될 수 있습니다.

오브젝트 이름
: 해석될 비규정화된 이름(보통 기존 별명의 이름)을 나타내는 문자 표현식. *오브젝트 이름*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0보다 크고 129자 미만이어야 합니다.

오브젝트-스키마
: 해석에 앞서 공급된 *objectname* 값을 규정화하는 데 사용된 스키마를 나타내는 문자 표현식. *오브젝트 스키마*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하며 길이는 0보다 크고 129자 미만이어야 합니다.

  *오브젝트-스키마*가 제공되지 않으면, 규정자에 기본 스키마가 사용됩니다.

함수 결과의 데이터 유형은 VARCHAR(128)입니다. *오브젝트 이름*이 널(NULL)이 될 수 있을 경우, 결과도 널(NULL)이 될 수 있습니다. 즉, *오브젝트 이름*이 널(NULL)이면 결과도 널(NULL) 값입니다. *오브젝트-스키마*가 널(NULL) 값인 경우, 기본 스키마 이름이 사용됩니다. 결과는 스키마 이름을 나타내는 문자열입니다. 결과 스키마는 다음 중 하나에 대한 스키마 이름을 표시할 수 있습니다.

테이블 *오브젝트 이름*의 값이, 스키마 이름이 리턴되는 테이블에 대해 해석된 별명이거나 테이블 이름(*오브젝트 이름*의 입력 또는 기본값이 리턴됨)이었습니다.

뷰 *오브젝트 이름*의 값이, 스키마 이름이 리턴되는 뷰에 대해 해석된 별명이거나 뷰 이름(*오브젝트 이름*의 입력 또는 기본값이 리턴됨)이었습니다.

정의되지 않은 오브젝트

*오브젝트 이름*의 값이, 스키마 이름이 리턴되는 정의되지 않은 오브젝트에 대해 해석된 별명이거나 정의되지 않은 오브젝트(*오브젝트-스키마*의 입력 또는 기본값이 리턴됨)이었습니다.

그러므로, 이 함수에 널(NULL)이 아닌 *오브젝트 이름* 값이 제공되면, 결과 스키마 이름을 갖고 있는 오브젝트 이름이 전혀 없을 경우에도 값이 항상 리턴됩니다. 예를 들어, TABLE_SCHEMA('DEPT', 'PEOPLE')는 카탈로그 항목이 발견되지 않을 경우에 'PEOPLE'을 리턴합니다.

예:

- PBIRD는 HEDGES.T1 테이블에 정의된 별명 PBIRD.A1을 사용하여 SYSCAT.TABLES로부터 제공된 테이블에 대한 통계를 선택하려고 합니다.

  ```
  SELECT NPAGES, CARD FROM SYSCAT.TABLES
    WHERE TABNAME = TABLE_NAME ('A1')
    AND TABSCHEMA = TABLE_SCHEMA ('A1')
  ```

  HEDGES.T1에 대해 요구되는 통계가 카탈로그에서 검색됩니다.

- HEDGES.X1을 사용하여 SYSCAT.TABLES로부터 HEDGES.X1이라고 하는 오브젝트에 대해 통계를 선택합니다. HEDGES.X1이 별명인지, 아니면 테이블인지 알 수 없으므로, TABLE_NAME 및 TABLE_SCHEMA를 사용합니다.

  ```
  SELECT NPAGES, CARD FROM SYSCAT.TABLES
    WHERE TABNAME = TABLE_NAME ('X1','HEDGES')
    AND TABSCHEMA = TABLE_SCHEMA ('X1','HEDGES')
  ```

  HEDGES.X1이 테이블이고 HEDGES.X1에 대해 요구된 통계가 카탈로그에서 검색된다고 가정하십시오.

- HEDGES.T2가 존재하지 않는 HEDGES.T2에 정의된 별명 PBIRD.A2를 사용하여 SYSCAT.TABLES로부터 제공된 테이블에 대한 통계를 선택합니다.

  ```
  SELECT NPAGES, CARD FROM SYSCAT.TABLES
    WHERE TABNAME = TABLE_NAME ('A2','PBIRD')
    AND TABSCHEMA = TABLE_SCHEMA ('A2',PBIRD')
  ```

  TABNAME = 'T2'이고 TABSCHEMA = 'HEDGES'인 일치되는 항목이 SYSCAT.TABLES에 없으면, 명령문은 0개의 레코드를 리턴시킵니다.

- 임의의 별명 항목에 대해 최종 참조 이름과 함께 SYSCAT.TABLES에서 각 항목의 규정화된 이름을 선택합니다.

```
SELECT TABSCHEMA AS SCHEMA,  TABNAME  AS NAME,
  TABLE_SCHEMA (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_SCHEMA,
  TABLE_NAME (BASE_TABNAME, BASE_TABSCHEMA)  AS REAL_NAME
  FROM SYSCAT.TABLES
```

  명령문은 카탈로그에 있는 각 오브젝트에 대한 규정화된 이름과, 별 이름 항목에 대한 최종 참조 이름(별명이 해석된 이후)을 리턴합니다. 별명이 아닌 모든 항목의 경우, BASE_TABNAME과 BASE_TABSCHEMA는 공백이므로, REAL_SCHEMA 및 REAL_NAME 컬럼에 널(NULL)이 들어 있게 됩니다.

TAN

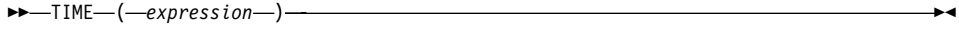
스키마는 SYSFUN입니다.

인수의 탄젠트를 리턴하며, 여기서 인수는 라디안으로 표시되는 각도입니다.

인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 함수가 처리할 수 있도록 배정밀도 부동 소수점 수로 변환해야 합니다.

함수의 결과는 배정밀도 부동 소수점 수입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

TIME



스키마는 SYSIBM입니다.

TIME 함수는 값에서 시간(time)을 리턴합니다.

인수는 시간, 시각 또는 CLOB도 LONG VARCHAR도 아닌 시간이나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 시간입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

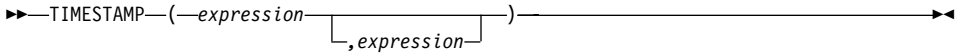
- 인수가 시간인 경우:
  - 결과는 해당 시간입니다.
- 인수가 시각인 경우:
  - 결과는 시각의 시간 부분입니다.
- 인수가 문자열(character string)인 경우:
  - 결과는 문자열에 의해 표시되는 시간입니다.

예:

- 당일(임의의 날) 현재 시간보다 최소한 1시간 늦게 수신된 모든 참고를 IN_TRAY 샘플 테이블에서 선택합니다.

```
SELECT * FROM IN_TRAY
  WHERE TIME(RECEIVED) >= CURRENT TIME  + 1 HOUR
```

TIMESTAMP



스키마는 SYSIBM입니다.

TIMESTAMP 함수는 하나의 값이나 값 쌍으로부터 시간소인(timestamp)을 리턴합니다.

인수에 대한 규칙은 초(second) 인수의 지정 여부에 따라 다릅니다.

- 하나의 인수만 지정할 경우:
  - 이것은 시간소인, 시간소인의 유효한 문자열, 표현 또는 CLOB도 아니고 LONG VARCHAR도 아닌 길이 14의 문자열이어야 합니다.

    길이 14의 문자열은 *yyyyxxddhhmmss* 양식으로 유효한 날짜와 시간을 나타내는 숫자 문자열이어야 합니다. 여기서, *yyyy*는 년도, *xx*는 월, *dd* 는 일, *hh*는 시간, *mm*은 분, 구*ss* 는 초를 나타냅니다.
- 두 인수를 모두 지정할 경우:
  - 첫번째 인수는 날짜 또는 날짜의 유효한 문자열 표현이고, 두 번째 인수는 시간 또는 시간의 유효한 문자열 표현이어야 합니다.

함수의 결과는 시간소인입니다. 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

다른 규칙은 초(second) 인수의 지정 여부에 따라 다릅니다.

- 두 인수를 모두 지정할 경우:
  - 결과는 첫번째 인수에 의해 지정된 날짜가 있는 시간소인과 두 번째 인수에 의해 지정된 시간입니다. 시간소인의 마이크로초 부분은 0입니다.
- 하나의 인수만 지정된 경우 이것은 시간소인입니다.
  - 결과는 해당 시간소인입니다.
- 하나의 인수만 지정된 경우 그것은 문자열입니다.

– 결과는 문자열에 의해 표시되는 시간소인입니다. 인수가 길이 14의 문자열 이면, 시간소인의 마이크로초 부분은 0입니다.

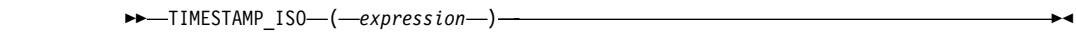
예:

- 컬럼 START_DATE(날짜)의 값이 1988-12-25이고, 컬럼 START_TIME(시간)의 값이 17.12.30인 것으로 가정합니다.

  ```
  TIMESTAMP(START_DATE, START_TIME)
  ```

  값 '1988-12-25-17.12.30.000000'을 리턴합니다.

TIMESTAMP_ISO



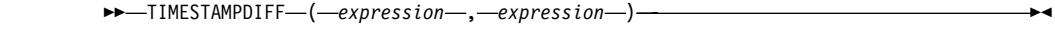
스키마는 SYSFUN입니다.

날짜, 시간 또는 시각 인수에 기초하여 시각 값을 리턴합니다. 인수가 날짜이면, 모든 시간 요소에 대해 0을 삽입합니다. 인수가 시간이면, 날짜 요소에 대해 CURRENT DATE의 값을 삽입하고 분수 시간 요소에 대해 0을 삽입합니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 TIMESTAMP입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

TIMESTAMPDIFF



스키마는 SYSFUN입니다.

두 시간소인의 차이에 따라 첫번째 인수가 정의한 유형의 측정된 간격 수를 리턴합니다.

첫번째 인수는 INTEGER 또는 SMALLINT가 될 수 있습니다. 간격(첫번째 인수)의 유효값은 다음과 같습니다.

1 초의 문

2 초

4 분

8 시간

16 일

32 주

64 월

128 분기

256 년

두 번째 인수는 두 개의 시간소인 유형을 빼서 그 결과를 CHAR(22)로 변환한 결과입니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

차이를 예측할 때 다음과 같은 사항을 가정할 수 있습니다.

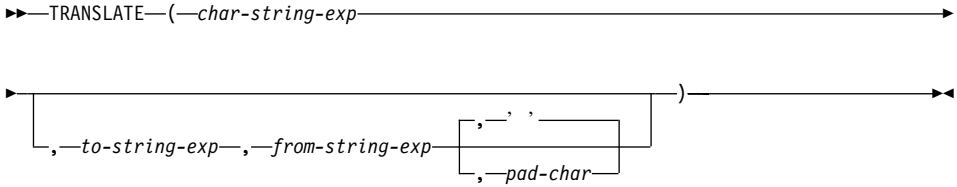
- 1년이 365일
- 1달이 30일
- 하루가 24시간
- 한시간이 60분

- 1분이 60초

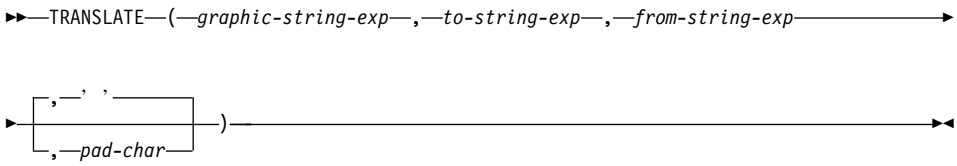
이러한 가정은 시간소인 기간에 두 번째 인수를 첫번째 인수에 지정된 간격 유형으로 변환할 때 사용됩니다. 리턴된 평가가 날 수에 따라 변할 수도 있습니다. 예를 들어, 날 수(간격 16)가 '1997-03-01-00.00.00'과 1997-02-01-00.00.00'에 대한 시간소인의 차이를 요구할 경우, 결과는 30입니다. 이것은 한 달을 30일로 가정하여 시간소인 사이의 차이를 한 달로 했기 때문입니다.

TRANSLATE

문자열 표현식:



그래픽 문자열 표현식:



스키마는 SYSIBM입니다.

TRANSLATE 함수는 문자 표현식의 하나 이상의 문자가 다른 문자로 변환되었을 수 있는 값을 리턴합니다.

함수의 결과는 첫번째 인수와 같은 데이터 유형과 코드 페이지를 갖습니다. 결과의 길이 속성은 첫번째 인수의 속성과 같습니다. 지 정된 표현식이 널(NULL)이 될 수 있으면, 결과는 널(NULL)이 될 수 있습니다. 지정된 표현식이 널(NULL)이면, 결과도 널(NULL)이 될 것입니다.

char-string-exp 또는 *graphic-string-exp*
변환될 문자열.

to-문자열-표현식
*문자열 표현식*의 특정 문자가 변환될 문자열입니다.

*to-string-exp*가 없고 데이터 유형이 그래픽이 아닐 경우, *char-string-exp*의 모든 문자는 대문자가 됩니다(즉, a - z 문자들이 A - Z 문자들로 변환되고, 발음 구별 부호가 있는 문자는 존재할 경우 해당되는 대문자로 변환될 것입니다. 예를 들어, 코드 페이지 850에서, é는 É에 맵핑되나, ÿ는 코드 페이지 850에 Ÿ가 포함되지 않으므로 맵핑되지 않습니다.).

from-문자열-표현식

*문자열-표현식*에서 발견될 경우, *to-문자열-표현식*의 해당 문자로 변환될 문자열입니다. *from-문자열-표현식*에 중복 문자가 포함되어 있으면, 발견되는 첫번째가 사용되고, 중복되는 것은 무시됩니다. *to-문자열-표현식*이 *from-문자열-표현식*보다 길면, 여분의 문자는 무시됩니다. *to-문자열-표현식*이 존재하면, *from-문자열-표현식*도 존재해야 합니다.

채움-문자-표현식

*to-문자열-표현식*이 *from-문자열-표현식*보다 짧은 경우 *to-문자열-표현식*을 채우는 데 사용할 단일 문자입니다. *채움-문자*는 길이 속성을 갖고 있어야 하고, 그렇지 않으면 오류가 발생합니다. 존재하지 않는 경우, 1바이트 공백으로 처리됩니다.

인수는 데이터 유형이 CHAR 또는 VARCHAR인 문자열이거나, 데이터 유형 GRAPHIC 또는 VARGRAPHIC의 그래픽 문자열이어야 합니다. 인수의 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB 또는 DBCLOB일 수 없습니다.

*그래픽-문자열-표현식*에서, *채움-문자*만 선택적이고(제공되지 않으면, 2바이트 공백으로 처리됨), 채움 문자를 포함하여 각 인수는 그래픽 데이터 유형이어야 합니다.

결과는 *to-문자열-표현식*의 해당 문자열에 대해 또는 어떤 해당 문자열도 존재하지 않을 경우, *채움-문자*에 의해 지정된 채움 문자에 대해 *from-문자열-표현식*에서 발생하는 *문자열-표현식* 또는 *그래픽-문자열-표현식*의 모든 문자열을 변환한 후에 발생하는 문자열입니다.

TRANSLATE 결과의 코드 페이지는 절대로 변환되지 않는 첫번째 피연산자의 코드 페이지와 같습니다. 각각의 다른 피연산자는 다른 피연산자 또는 첫번째 피연산자가 FOR BIT DATA로 정의(이 경우, 변환이 일어나지 않음)되어 있지 않으면, 첫번째 피연산자의 코드 페이지로 변환됩니다.

인수의 데이터 유형이 CHAR 또는 VARCHAR이면, *to-문자열-표현식* 및 *from-문자열-표현식*의 해당 문자는 동일한 바이트 수를 가져야 합니다. 예를 들면, 1바이트 문자에서 복수 바이트 문자로 또는 복수 바이트 문자에서 1바이트 문자로의 변환이 유효하지 않습니다. 이를 수행하려고 하면 오류가 야기됩니다. *채움-문자*는

유효한 복수 바이트 문자의 첫 바이트가 되지 않아야 하며, SQLSTATE 42815가 리턴됩니다. *채움-문자*가 제시되지 않으면, 1바이트 공백으로 처리됩니다.

*문자열 표현식*만 지정되면, 1바이트 문자는 단일 문자형(monocased)이 되고 2바이트 문자는 변경되지 않고 남아 있습니다.

예:

- 호스트 변수 SITE(VARCHAR(30))가 'Hanauma Bay'의 값을 갖는다고 가정합니다.

  ```
  TRANSLATE(:SITE)
  ```

  값 'HANAUMA BAY'를 리턴합니다.

  ```
  TRANSLATE(:SITE 'j','B')
  ```

  값 'Hanauma jay'을 리턴합니다.

  ```
  TRANSLATE(:SITE,'ei','aa')
  ```

  값 'Heneume Bey'를 리턴합니다.

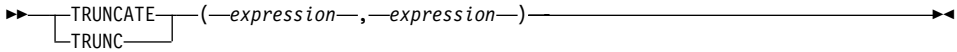
  ```
  TRANSLATE(:SITE,'bA','Bay','%')
  ```

  값 'HAnAumA bA%'를 리턴합니다.

  ```
  TRANSLATE(:SITE,'r','Bu')
  ```

  값 'Hana ma ray'를 리턴합니다.

TRUNCATE 또는 TRUNC



스키마는 SYSFUN입니다.

소수점의 오른쪽 인수2 자리로 절단된 인수1을 리턴합니다. 인수2가 음수이면, 인수1은 소수점 왼쪽으로 인수2 자리의 절대 값으로 절단됩니다.

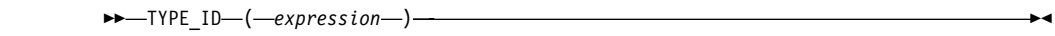
첫번째 인수는 내장형 숫자 데이터 유형이면 어느 것이나 가능합니다. 두번째 인수는 INTEGER 또는 SMALLINT여야 합니다. 함수가 처리할 수 있도록 DECIMAL과 REAL은 배정밀도 부동 소수점 수로 변환됩니다.

함수의 결과는 다음과 같습니다.

- 첫번째 인수가 INTEGER 또는 SMALLINT일 경우 INTEGER입니다.
- 첫번째 인수가 BIGINT일 경우 BIGINT입니다.
- 첫번째 인수가 DECIMAL 또는 DOUBLE일 경우 DOUBLE입니다.

결과가 널(NULL)이 될 수 있습니다. 즉, 널(NULL)인 인수가 있을 경우 결과도 널(NULL)입니다.

TYPE_ID



스키마는 SYSIBM입니다.

TYPE_ID 함수는 *표현식*의 동적 데이터 유형 내부 유형 식별자를 리턴합니다.

인수는 사용자 정의 구조 유형입니다. [48]

함수 결과의 데이터 유형은 INTEGER입니다. *표현식*이 널(NULL)이 될 수 있을 경우, 결과도 널(NULL)이 될 수 있습니다. *표현식*이 널(NULL)인 경우 결과가 널(NULL) 값입니다.

TYPE_ID 함수에 의해 리턴되는 값은 데이터베이스를 통해 이식가능(portable)하지 않습니다. 동적 데이터 유형의 유형 이름과 유형 스키마가 같아도 그 값은 서로 다를 수 있습니다. 이식성을 위해 코딩할 경우, TYPE_SCHEMA 및 TYPE_NAME 함수를 사용하여 유형 스키마와 유형 이름을 판별하십시오.

예:

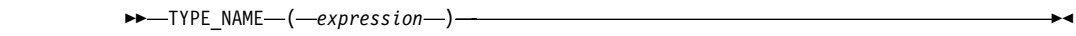
- 테이블 계층에는 EMP 유형인 루트 테이블 EMPLOYEE와 MGR 유형인 서브테이블 MANAGER이 있습니다. 또다른 테이블 ACTIVITIES에는 REF(EMP) SCOPE EMPLOYEE로 정의된 WHO_RESPONSIBLE 컬럼이 포함됩니다. ACTIVITIES에서의 각 참조에 대해, 참조에 대응하는 행의 내부 유형 식별자를 표시하십시오.

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
         TYPE_ID(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  DEREF 함수를 사용하여 행에 따라 오브젝트를 리턴합니다.

48. 사용자 정의 함수(UDF) 작성시, 이 함수를 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 모든 구조화된 데이터 유형을 허용하므로, 다른 사용자 정의 유형을 지원하기 위해 별도의 시그니처를 작성할 필요가 없습니다.

TYPE_NAME



스키마는 SYSIBM입니다.

TYPE_NAME 함수는 *표현식*의 동적 데이터 유형의 규정되지 않은 이름을 리턴합니다.

인수는 사용자 정의 구조 유형입니다. [49]

함수 결과의 데이터 유형은 VARCHAR(18)입니다. *표현식*이 널(NULL)이 될 수 있을 경우, 결과도 널(NULL)이 될 수 있습니다. *표현식*이 널(NULL)인 경우 결과가 널(NULL) 값입니다. TYPE_SCHEMA 함수를 사용하여 TYPE_NAME에 의해 리턴되는 유형 이름의 스키마 이름을 지정하십시오.

예:

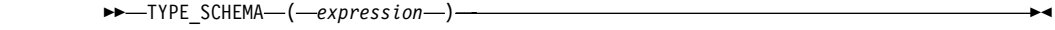
- 테이블 계층에는 EMP 유형인 루트 테이블 EMPLOYEE와 MGR 유형인 서브테이블 MANAGER이 있습니다. 또다른 테이블 ACTIVITIES에는 REF(EMP) SCOPE EMPLOYEE로 정의된 WHO_RESPONSIBLE 컬럼이 포함됩니다. ACTIVITIES에서의 각 참조에 대해, 참조에 대응하는 행의 유형을 표시하십시오.

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
         TYPE_NAME(DEREF(WHO_RESPONSIBLE)),
         TYPE_SCHEMA(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  DEREF 함수를 사용하여 행에 따라 오브젝트를 리턴합니다.

49. 사용자 정의 함수(UDF) 작성시, 이 함수를 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 모든 구조화된 데이터 유형을 허용하므로, 다른 사용자 정의 유형을 지원하기 위해 별도의 시그니처를 작성할 필요가 없습니다.

TYPE_SCHEMA



스키마는 SYSIBM입니다.

TYPE_SCHEMA 함수는 *표현식*의 동적 데이터 유형의 스키마 이름을 리턴합니다.

인수는 사용자 정의 구조 유형입니다. [50]

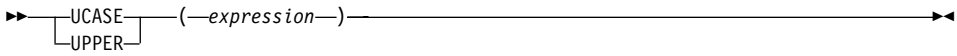
함수 결과의 데이터 유형은 VARCHAR(128)입니다. *표현식*이 널(NULL)이 될 수 있을 경우, 결과도 널(NULL)이 될 수 있습니다. *표현식*이 널(NULL)인 경우 결과가 널(NULL) 값입니다. TYPE_NAME 함수를 사용하여 TYPE_SCHEMA에 의해 리턴되는 스키마 이름과 연관된 유형 이름을 결정하십시오.

예:

417 페이지의 『TYPE_NAME』에 있는 예 부분을 참조하십시오.

50. 사용자 정의 함수(UDF) 작성시, 이 함수를 소스 함수로 사용할 수 없습니다. 이 함수는 인수로 모든 구조화된 데이터 유형을 허용하므로, 다른 사용자 정의 유형을 지원하기 위해 별도의 시그니처를 작성할 필요가 없습니다.

UCASE 또는 UPPER

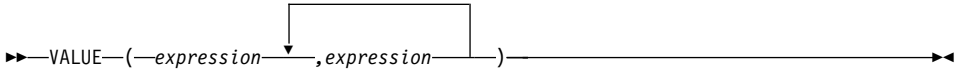


스키마는 SYSIBM입니다.[51]

UCASE 또는 UPPER 함수는 첫번째 인수(*char-string-exp*)만 지정되는 것을 제외하고 TRANSLATE 함수와 같습니다. 412 페이지의 『TRANSLATE』에서 자세한 정보를 참조하십시오.

51. 상향 호환성을 위해 이 함수의 SYSFUN 버전이 계속 사용 가능합니다. 버전 5 문서에서 자세한 내용을 참조하십시오.

VALUE



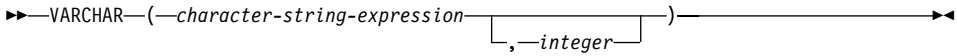
스키마는 SYSIBM입니다.

VALUE 함수는 널(NULL)이 아닌 첫번째 인수를 리턴합니다.

VALUE는 COALESCE의 동의어입니다. 299 페이지의 『COALESCE』에서 자세한 내용을 참조하십시오.

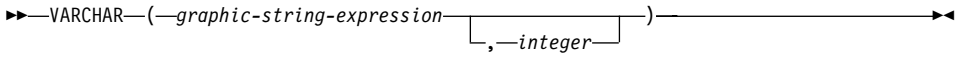
VARCHAR

문자 대 Varchar:



날짜시간 대 Varchar:

그래픽 대 Varchar:



스키마는 SYSIBM입니다.

VARCHAR 함수는 문자열, 날짜시간 값 또는 그래픽 문자열(UCS-2)의 가변 길이 문자열 표현을 리턴합니다.

함수의 결과는 가변 길이 문자열입니다(VARCHAR 데이터 유형). 첫번째 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, 첫번째 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

Graphic 대 Varchar는 UCS-2 데이터베이스에만 유효합니다. 비 유니코드 데이터베이스의 경우에는 이것이 허용되지 않습니다.

문자 대 Varchar

문자열 표현식

값이 LONG VARGRAPHIC 및 DBCLOB 외의 문자열 데이터 유형이어야 하고, 최대 길이가 32 672바이트여야 하는 표현식.

정수

결과로 나오는 가변 길이 문자열에 대한 길이 속성 값은 0과 32 672 사이여야 합니다. 이 인수를 지정하지 않으면, 결과의 길이는 인수의 길이와 같습니다.

날짜시간 대 Varchar

날짜시간 표현식

값이 날짜, 시간 또는 시각 데이터 유형이어야 하는 표현식입니다.

Graphic 대 Varchar

그래픽 문자열 표현식

값이 LONG VARGRAPHIC 또는 DBCLOB 외의 그래픽 문자열 데이터 유형이어야 하고 최대 길이가 16 336바이트여야 하는 표현식.

정수

결과로 나오는 가변 길이 문자열에 대한 길이 속성 값은 0과 32 672 사이여야 합니다. 이 인수를 지정하지 않으면, 결과의 길이는 인수의 길이와 같습니다.

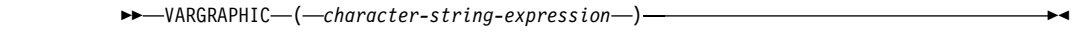
예:

- EMPLOYEE 테이블을 사용하여 호스트 변수 JOB_DESC(VARCHAR(8))를 사원 Delores Quintana의 직업 설명(CHAR(8)로 정의된 JOB)에 해당하는 VARCHAR로 설정합니다.

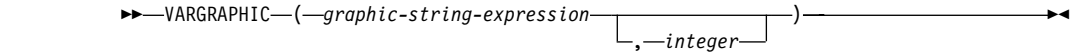
```
SELECT VARCHAR(JOB)
  INTO :JOB_DESC
  FROM EMPLOYEE
  WHERE LASTNAME = 'QUINTANA'
```

VARGRAPHIC

문자 대 Vargraphic:



그래픽 대 Vargraphic:



스키마는 SYSIBM입니다.

VARGRAPHIC 함수는 다음의 그래픽 문자열 표현을 리턴합니다.

- 1바이트 문자를 2바이트 문자로 변환하는 문자열 값입니다.
- 첫번째 인수가 그래픽 문자열의 유형일 경우, 그래픽 문자열 값입니다.

함수의 결과는 가변 길이 그래픽 문자열 입니다(VARGRAPHIC data type). 첫번째 인수가 널(NULL)이 될 수 있으면 결과도 널(NULL)이 될 수 있습니다. 즉, 첫번째 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

그래픽 대 Vargraphic

문자열 표현식

값이 LONG VARCHAR 또는 CLOB 외의 다른 문자열 데이터 유형이어야 하고 최대 길이가 16 336 바이트이하여야 하는 표현식.

결과의 길이 속성은 인수의 길이 속성과 같아야 합니다.

S는 *문자열 표현식*의 값을 나타낸다고 합시다. S의 각 1바이트 문자는 결과에서 해당되는 2바이트 표시로 변환되거나 2바이트 대체 문자로 변환됩니다. S에서 각 2바이트 문자는 '그대로' 대응됩니다. 2바이트 문자의 첫번째 바이트가 S의 마지막 바이트로 나타나면, *그것*은 2바이트 대체 문자로 변환됩니다. S 내에서 문자의 순차적인 순서는 보유됩니다.

다음은 변환에 대한 추가 고려사항입니다.

- 유니코드 데이터베이스의 경우, 이 함수는 문자열을 피연산자의 코드 페이지에서 UCS-2로 변환합니다. DBCS 문자를 포함하여 피연산자의 모든 문자가 변환됩니다. 두 번째 인수가 주어지면, 결과 UCS-2 문자열의 원하는 길이(UCS-2 문자의 수)를 지정합니다.
- VARGRAPHIC 함수를 사용한 2바이트 코드값으로의 변환은 피연산자의 코드 페이지에 따라 달라집니다.
- 피연산자의 2바이트 문자는 변환되지 않습니다(예외는 1505 페이지의 『부록O. 일본어 및 대만어의 EUC 고려사항』 참조). 기타 모든 문자는 해당 2바이트 설명(depiction)으로 변환됩니다. 해당되는 2바이트 설명이 없으면, 코드 페이지의 2바이트 대체 문자가 사용됩니다.
- 하나 이상의 2바이트 대체 문자가 결과로 리턴되면 경고나 오류 코드가 생성되지 않습니다.

그래픽 대 Vargraphic

그래픽 문자열 표현식

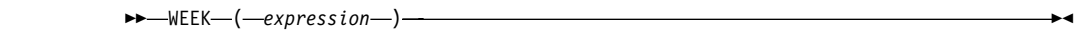
그래픽 문자열인 값을 리턴하는 표현식.

정수

결과로 나오는 가변 길이 문자열에 대한 길이 속성입니다. 값은 0과 16 336 사이여야 합니다. 이 인수를 지정하지 않으면, 결과의 길이는 인수의 길이와 같습니다.

*그래픽-문자열-표현식*의 길이가 결과의 길이 속성보다 길면 그 길이는 절단되고, 절단되는 문자가 모두 공백이 아니고 *그래픽 문자열 표현식*이 long 문자열(LONG VARGRAPHIC 또는 DBCLOB)이면, 경고가 표시됩니다(SQLSTATE 01004).

WEEK



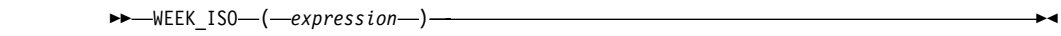
스키마는 SYSFUN입니다.

인수의 년도에 있는 주(week)를 1-54 범위내의 정수 값으로 리턴합니다. 주는 일요일부터 시작합니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

WEEK_ISO



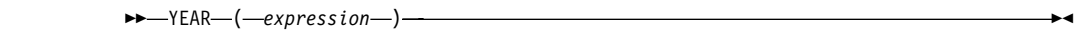
스키마는 SYSFUN입니다.

인수의 년도에 있는 주(week)를 1-53 범위내의 정수 값으로 리턴합니다. 주는 월요일부터 시작합니다. 주 1은 목요일을 포함하는 년도의 첫번째 주입니다. 이것은 1월 4일을 포함하는 첫번째 주와 같습니다.

인수는 날짜, 시각 또는 CLOB도 아니고 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 INTEGER입니다. 결과가 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)일 경우, 결과도 널(NULL) 값입니다.

YEAR



스키마는 SYSIBM입니다.

YEAR 함수는 값의 년도 부분을 리턴합니다.

인수는 날짜, 시각, 날짜 기간, 시각 기간 또는 CLOB도 LONG VARCHAR도 아닌 날짜나 시각의 유효한 문자열 표현이어야 합니다.

함수의 결과는 큰 정수입니다. 인수가 널(NULL)이 될 수 있으면, 결과도 널(NULL)이 될 수 있습니다. 즉, 인수가 널(NULL)이면 결과도 널(NULL) 값입니다.

지정된 인수의 데이터 유형에 따라 다른 규칙이 적용됩니다.

- 인수가 날짜, 시간소인 또는 날짜나 시간소인의 유효한 문자열 표현인 경우:
  - 결과는 값의 년도 부분으로서, 1에서 9 999 사이의 정수입니다.
- 인수가 날짜 기간이거나 시각 기간인 경우:
  - 결과는 값의 연도 부분으로서, -9 999에서 9 999 사이의 정수입니다. 0이 아닌 결과는 인수와 같은 부호를 갖습니다.

예:

- 같은 달력 년도에서 시작(PRSTDATE)되고 종료(PRENDATE)되도록 스케줄된 PROJECT 테이블의 모든 프로젝트를 선택합니다.

```
SELECT * FROM PROJECT
  WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```

- 완료하는 데 1년 이하가 소요되도록 스케줄된 모든 프로젝트를 PROJECT 테이블에서 선택합니다.

```
SELECT * FROM PROJECT
  WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

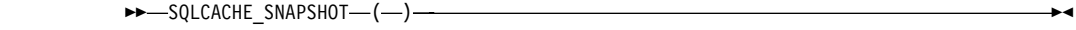
테이블 함수

테이블 함수는 명령문의 FROM절에서만 사용될 수 있습니다. 그러나, 표현식이나 컬럼 함수는 테이블 함수 내에서 사용될 수 없습니다.

테이블 함수는 간단한 CREATE TABLE문으로 작성된 테이블을 닮아 테이블의 컬럼을 리턴합니다.

다음에 오는 테이블 함수는 스키마 이름으로 규정화될 수 있습니다.

SQLCACHE_SNAPSHOT



스키마는 SYSFUN입니다.

SQLCACHE_SNAPSHOT는 DB2 동적 SQL문 캐쉬의 스냅샷 결과를 리턴합니다.

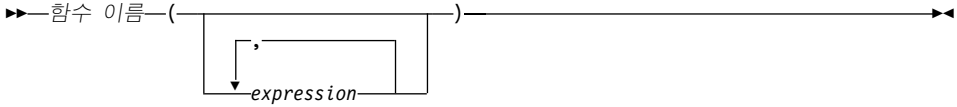
함수는 어떠한 인수도 취하지 않습니다.

아래 나열된 테이블을 리턴합니다. 컬럼에 대한 *시스템 모니터 안내 및 참조서*에서 자세한 내용을 참조하십시오.

표 17. SQLCACHE_SNAPSHOT 테이블 함수에 의해 리턴된 테이블의 컬럼 이름 및 데이터 유형

컬럼 이름	데이터 유형
NUM_EXECUTIONS	INTEGER
NUM_COMPILATIONS	INTEGER
PREP_TIME_WORST	INTEGER
PREP_TIME_BEST	INTEGER
INT_ROWS_DELETED	INTEGER
INT_ROWS_INSERTED	INTEGER
ROWS_READ	INTEGER
INT_ROWS_UPDATED	INTEGER
ROWS_WRITE	INTEGER
STMT_SORTS	INTEGER
TOTAL_EXEC_TIME_S	INTEGER
TOTAL_EXEC_TIME_MS	INTEGER
TOT_U_CPU_TIME_S	INTEGER
TOT_U_CPU_TIME_MS	INTEGER
TOT_S_CPU_TIME_S	INTEGER
TOT_S_CPU_TIME_MS	INTEGER
DB_NAME	VARCHAR(8)
STMT_TEXT	CLOB(64K)

사용자 정의 함수



사용자 정의 함수는 SQL 언어의 기존 내장 함수에 대한 확장 또는 추가사항입니다. 사용자 정의 함수(UDF)는 호출될 때마다 단일 값을 리턴하는 스칼라 함수, 유사 값 세트를 제공하고 그 세트에 대한 단일 값을 리턴하는 컬럼 함수, 한 행을 리턴하는 행 함수, 또는 하나의 테이블을 리턴하는 테이블 함수가 될 수 있습니다. UDF는 기존 컬럼 함수에서 전래될 경우에만 컬럼 함수가 될 수 있습니다.

내장 함수가 나타날 수 있는 동일한 문맥에서 데이터베이스에 등록된 사용자 정의 스칼라 함수 또는 컬럼 함수를 참조할 수 있습니다.

데이터베이스에 등록된 사용자 정의 테이블 함수는 440 페이지의 『from절』에 기술된 것과 같이 SELECT의 FROM절에서만 참조할 수 있습니다.

사용자 정의 행 함수는 사용자 정의 유형에 대한 변환 함수로 등록된 경우에 내재적으로만 참조될 수 있습니다.

사용자 정의 함수는 규정화되거나 규정화되지 않은 함수 이름에 의해 참조됩니다. 그 뒤에 함수 인수(있는 경우)를 묶는 괄호가 옵니다.

함수의 인수는 숫자와 위치면에서 데이터베이스에 등록된 사용자 정의 함수에 지정된 매개변수에 일치해야 합니다. 또한, 인수는 정의된 해당 매개변수의 데이터 유형으로 이전할 수 있는 데이터 유형으로 되어 있어야 합니다 (653 페이지의 『CREATE FUNCTION』 참조).

함수의 결과는 사용자 정의 함수(UDF)가 등록될 때 지정된 RETURNS절에서 지정된 대로입니다. RETURNS절은 함수가 테이블 함수인지의 여부를 판별합니다.

함수가 등록될 때 RETURNS NULL ON NULL INPUT 절이 지정되거나 기본값으로 설정된 경우, 인수가 널(NULL)이면, 결과는 널입니다. 테이블 함수의 경우, 이것은 행이 없는 리턴 테이블(공백 테이블)을 의미하는 것으로 해석됩니다.

SYSFUN 스키마 내에 제공되는 사용자 정의 함수의 콜렉션이 있습니다(240 페이지의 표15 참조).

예:

- 스크립트 형식 이력서로부터 집 주소를 추출하기 위해 ADDRESS라고 하는 스칼라 함수를 작성하였다고 가정합시다. ADDRESS 함수에는 CLOB 인수가 입력되어 VARCHAR(4000)을 리턴합니다. 다음 예는 ADDRESS 함수의 호출을 보여줍니다.

  ```
  SELECT EMPNO, ADDRESS(RESUME) FROM EMP_RESUME
    WHERE RESUME_FORMAT = 'SCRIPT'
  ```

- 숫자 컬럼 A가 있는 테이블 T2와 이전 예에서 설명된 ADDRESS 함수를 가정합니다. 다음 예에서는 틀린 인수를 이용하여 ADDRESS 함수를 호출하려는 시도를 보여 줍니다.

  ```
  SELECT ADDRESS(A) FROM T2
  ```

  인수에서 이전할 수 있는 매개변수 및 일치하는 이름을 갖는 함수가 없으므로 오류(SQLSTATE 42884)가 발생합니다.

- 명령문이 실행될 때 사용중이었던 서버 머신상의 세션 관련 정보를 리턴하기 위한 테이블 함수 WHO가 작성되었다고 가정합시다. 다음은 FROM절에서 WHO를 호출한 예입니다(필수 상관 이름을 갖는 TABLE 키워드).

  ```
  SELECT ID, START_DATE, ORIG_MACHINE
  FROM TABLE( WHO() ) AS QQ
  WHERE START_DATE LIKE 'MAY%'
  ```

  WHO() 테이블의 컬럼 이름은 CREATE FUNCTION문에서 정의됩니다.

제5장 조회

*조회(query)*는 결과 테이블을 지정합니다.

조회는 특정 SQL문의 구성요소입니다. 조회 양식은 세 가지입니다.

- 부속 선택
- fullselect
- 선택 명령문

1136 페이지의 『SELECT INTO』에서 설명된 많아야 하나의 행을 검색하는 데 사용할 수 있는 또다른 SQL문이 있습니다.

권한 부여(Authorization)

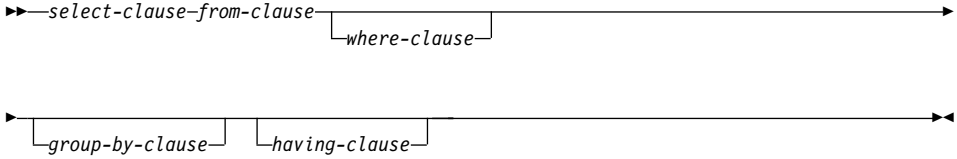
조회에서 참조되는 각 테이블 또는 뷰에 대해, 명령문의 권한 부여 ID는 최소한 다음 중 하나를 갖고 있어야 합니다.

- SYSADM 또는 DBADM 권한
- CONTROL 특권
- SELECT 특권

정적 SQL문에 포함된 조회에 대해서는 그룹 특권이 검사되지 않습니다.

조회에서 참조된 별명(nickname)의 경우에는, 연합 서버에서 고려될 특권이 없습니다. 별명(nickname)으로 참조된 테이블이나 뷰에 대한 데이터 소스의 권한 부여 요건은 조회가 처리될 때 적용됩니다. 명령문의 권한 부여 ID는 서로 다른 원격 권한 부여 ID로 맵핑될 수도 있습니다.

부속 선택



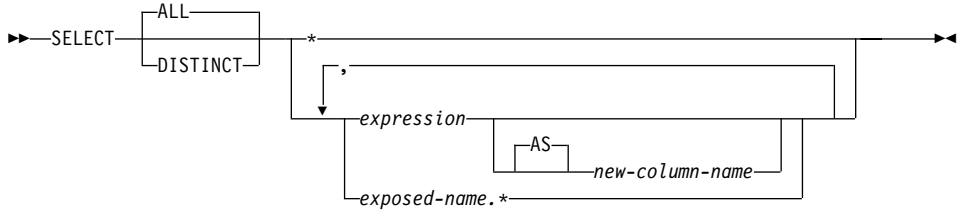
*부속 선택*은 fullselect의 구성요소입니다.

부속 선택은 FROM 절에서 식별된 테이블, 뷰 또는 별명에서 파생되는 결과 테이블을 지정합니다. 파생(derivation)은 각 작업의 결과가 다음 작업에 대한 입력이 되는 일련의 작업 순서로 설명됩니다(이것은 부속 선택을 설명하는 유일한 방법입니다.). 파생을 수행하는 데 사용되는 방법은 이 설명과 많이 다를 수 있습니다.

부속 선택의 절은 다음 순서로 처리됩니다.

1. FROM절
2. WHERE절
3. GROUP BY절
4. HAVING절
5. SELECT절

select절



SELECT절은 최종 결과 테이블의 컬럼들을 지정합니다. 컬럼값은 R에 대해 *선택 목록*의 응용프로그램에 의해 생성됩니다. 선택 목록은 SELECT절에 지정된 이름이나 표현식이고, R은 부속 선택의 이전 작업에 대한 결과입니다. 예를 들어, 지정되는 유일한 절이 SELECT, FROM 및 WHERE이고, R은 WHERE절의 결과입니다.

ALL

최종 결과 테이블의 모든 행을 보유하고, 중복사항을 제거하지 않습니다. 이것은 기본값입니다.

DISTINCT

최종 결과 테이블의 각 중복 행 집합중 대부분을 제거합니다. DISTINCT를 사용할 경우, 결과 테이블의 어떤 문자열 컬럼도 최대 255 바이트보다 큰 길이를 가질 수 없으며, 어떤 컬럼도 LONG VARCHAR, LONG VARGRAPHIC, DATALINK, LOB 유형, 이 유형들 중 하나에 대한 구별 유형, 또는 구조화 유형이 될 수 없습니다. DISTINCT는 부속 선택에서 두 번 이상 사용될 수 있습니다. 여기에는 SELECT DISTINCT, 선택 목록의 컬럼 함수에서의 사용, 그리고 부속 선택의 부속 조회가 포함됩니다.

첫번째의 각 값이 두번째의 해당 값과 같을 경우에만 두 행은 다른 행과 중복됩니다. 중복이라고 판별되면, 두 개의 널(NULL) 값은 같은 것으로 간주됩니다.

선택 목록 표기법:

* 테이블 R의 컬럼을 식별하는 이름의 목록을 나타냅니다. 목록의 첫번째 이름은 R의 첫번째 컬럼을 식별하며, 두번째 이름은 R의 두번째 컬럼을 식별하고, 등등입니다.

이름 목록은 SELECT절을 포함하는 프로그램이 바인드될 때 설정됩니다. 그러므로, *(별표)는 테이블 참조를 포함하는 명령문의 바인드 후에 테이블에 추가된 컬럼은 식별하지 않습니다.

표현식

결과 컬럼의 값을 지정합니다. 『제3장 언어 요소』에는 모든 유형의 표현식이 기술되지만, 대체로 컬럼 이름이 포함된 표현식이 사용됩니다. 선택 목록에서 사용되는 각 컬럼 이름은 R의 컬럼을 정확하게 식별해야 합니다.

새 컬럼 이름 또는 **AS** *새 컬럼 이름*

결과 컬럼을 명명하거나 재명명합니다. 이름이 규정화되어 있거나 고유해서는 안 됩니다. 컬럼 이름의 후속 사용은 다음으로 제한됩니다.

- AS절에 지정된 새 컬럼 이름은 이름이 고유할 경우 order-by절에서 사용할 수 있습니다.
- 선택 목록의 AS절에 지정된 새 컬럼 이름은 부속 선택 내의 다른 절(where절, group-by절 또는 having절)에서 사용할 수 없습니다.
- AS절에 지정된 새 컬럼 이름은 update절에서 사용할 수 없습니다.
- AS절에 지정된 새 컬럼 이름은 내포된 테이블 표현식의 fullselect, 공통 테이블 표현식, 그리고 CREATE VIEW 밖에서 알려져 있습니다.

*이름.**

*표시 이름*으로 식별되는 결과 테이블의 컬럼을 식별하는 이름 목록을 나타냅니다. *표시 이름*은 테이블 이름, 뷰 이름, 별명(nickname) 또는 상관 이름이 될 수 있으며, FROM 절에서 명명된 테이블, 뷰 또는 별명(nickname)을 지시해야 합니다. 목록의 첫번째 이름은 테이블, 뷰 또는 별명(nickname)의 첫번째 컬럼을 식별하고, 목록의 두 번째 이름은 테이블, 뷰 또는 별명(nickname)의 두 번째 컬럼을 식별합니다.

이름 목록은 SELECT절을 포함하는 명령문이 바인드될 때 설정됩니다. 그러므로, *는 명령문이 바인드된 후에 테이블에 추가된 컬럼은 식별하지 않습니다.

SELECT 결과에 있는 컬럼의 수는 선택 목록(즉, 명령문이 준비될 때 설정된 목록)의 작동 양식으로 된 표현식의 수와 동일하며 500을 초과할 수 없습니다.

문자열 컬럼에서의 한계

선택 목록에서의 제한사항에 대해서는 92 페이지의 『가변 길이 문자열 사용시 제한사항』에서 참조하십시오.

선택 목록 적용

R에 선택 목록을 적용한 결과의 일부는 GROUP BY나 HAVING 중 어느 것이 사용되는 지에 따라 다릅니다. 결과는 두 개의 별개 목록으로 설명됩니다.

GROUP BY 또는 HAVING이 사용된 경우:

- 선택 목록에 사용된 표현식 *X*(컬럼 함수가 아님)는 다음 경우에 GROUP BY 절을 포함해야 합니다.
  - 각 컬럼 이름이 R의 컬럼을 명확하게 나타내는 *그룹 표현식*(449 페이지의 『group-by절』 참조) 또는
  - *X*에서 별개의 *그룹 표현식*으로 참조된 R의 각 컬럼입니다.
- 선택 목록은 R의 각 그룹에 적용되며, 결과는 R에 있는 그룹 만큼의 행을 포함합니다. 선택 목록이 R의 그룹에 적용될 때에는 그 그룹이 선택 목록에 있는 컬럼 함수의 인수 소스입니다.

GROUP BY나 HAVING이 둘다 사용되지 않을 경우:

- 선택 목록에 어떤 컬럼 함수도 포함되지 않거나 선택 목록의 각 *컬럼 이름*이 컬럼 함수 내에 지정되어야 하며 또는 상관 컬럼 함수여야 합니다.
- 선택에 컬럼 함수가 포함되지 않으면, 선택 목록은 R의 각 행에 적용되고, 결과에는 R에 있는 것만큼의 행들이 포함됩니다.
- 선택 목록이 컬럼 함수의 목록이면, R은 함수에 대한 인수의 소스이고, 선택 목록을 적용하는 결과는 한 행입니다.

결과의 *n*번째 컬럼에는 선택 목록의 작동(operational) 양식에서 *n*번째 표현식을 적용함으로써 지정된 값이 포함됩니다.

결과 컬럼의 널(NULL) 속성: 결과 컬럼이 다음으로부터 도출된 경우, 그 결과 컬럼은 널(NULL) 값을 허용하지 않습니다.

- 널(NULL) 값을 허용하지 않는 컬럼
- 상수

- COUNT 또는 COUNT_BIG 함수
- 표시기 변수를 갖고 있지 않은 호스트 변수
- 널(NULL) 값을 허용하는 피연산자를 포함하지 않는 스칼라 함수나 표현식.

결과 컬럼이 다음으로부터 도출된 경우, 그 결과 컬럼은 널(NULL) 값을 허용합니다.

- COUNT 또는 COUNT_BIG를 제외한 모든 컬럼 함수
- 널(NULL) 값을 허용하는 컬럼
- 널(NULL)을 허용하는 피연산자를 포함하는 스칼라 함수 또는 표현식
- 같은 값을 포함하는 인수가 있는 NULLIF 함수
- 표시기 변수를 갖고 있는 호스트 변수
- 선택 목록에 있는 해당 항목 중 최소한 하나가 널(NULL)일 가능성이 있을 때 설정 작업의 결과
- 산술식에서 도출되고, DFT_SQLMATHWARN을 yes로 설정하여 데이터베이스를 구성한 산술식 또는 뷰 컬럼.
- 참조해제(dereference) 연산.

결과 컬럼의 이름:

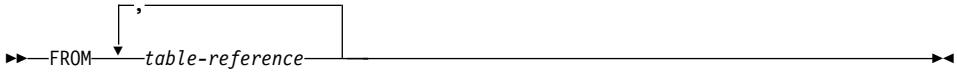
- AS절이 지정된 경우, 결과 컬럼의 이름은 AS절에 지정된 이름입니다.
- AS절이 지정되지 않고 결과 컬럼이 컬럼으로부터 도출되지 않은 경우, 결과 컬럼 이름은 그 컬럼의 규정화되지 않은 이름입니다.
- AS절이 지정되지 않고 결과 컬럼이 참조해제(deference operation) 연산을 사용하여 도출되는 경우, 결과 컬럼 이름은 참조해제 연산 목표 컬럼의 규정화되지 않은 이름입니다.
- 다른 모든 결과 컬럼 이름은 명명되지 않습니다. [52]

결과 컬럼의 데이터 유형: SELECT에 대한 결과의 각 컬럼은 도출되는 표현식으로부터 데이터 유형을 획득합니다.

52. 시스템은 임시 숫자를 이 컬럼에 지정합니다(문자열로서).

표현식이 다음과 같을 경우:	결과 컬럼의 데이터 유형:
숫자 컬럼의 이름	DECIMAL 컬럼에 대해 동일한 정밀도와 스케일을 갖는, 컬럼의 데이터 유형과 같음.
정수 상수	INTEGER
십진 상수	DECIMAL, 상수의 정밀도와 스케일
부동 소수점 상수	DOUBLE
숫자 변수 이름	DECIMAL 변수에 대해 동일한 정밀도와 스케일을 갖는, 변수의 데이터 유형과 같음.
표현식	데이터 유형 속성에 대한 182 페이지의『표현식』에서 설명을 참조하십시오.
임의의 함수	(결과의 데이터 유형을 판별하려면『제4장 함수』참조.)
n바이트를 나타내는 16진 상수	VARCHAR(n). 코드 페이지는 데이터베이스 코드 페이지입니다.
문자열 컬럼의 이름	같은 길이 속성을 갖고 있는, 컬럼의 데이터 유형과 같음.
문자열 변수의 이름	같은 길이 속성을 갖고 있는, 변수의 데이터 유형과 같음. 변수의 데이터 유형은 SQL 데이터 유형과 같지 않으므로(예를 들어, C에서 NUL 종결 문자열), 결과 컬럼은 길이가 다양한 문자열입니다.
길이 *n*의 리터럴	VARCHAR(*n*)
길이 *n*의 그래픽 리터럴	VARGRAPHIC(*n*)
날짜 시간 컬럼의 이름	컬럼의 데이터 유형과 같음.
사용자 정의 유형 컬럼의 이름	컬럼의 데이터 유형과 같음.
참조 유형 컬럼의 이름	컬럼의 데이터 유형과 같음.

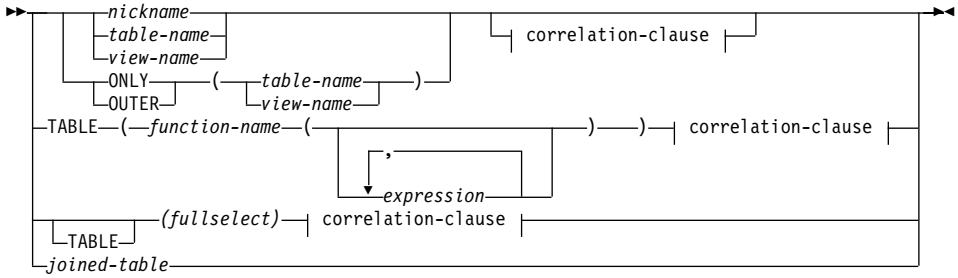
from절



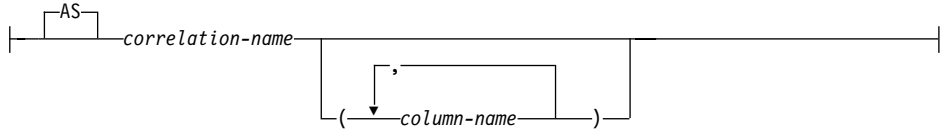
FROM절은 중간 결과 테이블을 지정합니다.

하나의 테이블 참조가 지정되면, 중간 결과 테이블은 단순히 그 테이블 참조이 결과가 됩니다. 하나 이상의 테이블 참조가 지정된 경우, 중간 결과 테이블은 지정된 테이블 참조의 가능한 모든 행의 조합으로 구성됩니다(데카르트곱). 결과의 각 행은 두 번째 테이블 참조로부터의 행과 병합된 첫번째 테이블 참조로부터의 행이고, 다시 세번째 테이블 참조로부터의 행과 병합되어 있습니다. 결과의 행 수는 개개의 모든 테이블 참조 행 수의 곱입니다. *테이블 참조*에 대한 441 페이지의 『테이블 참조』에서 내용을 참조하십시오.

테이블 참조



correlation-clause:



테이블 참조로 지정된 각 *테이블 이름*, *뷰 이름* 또는 *별명(nickname)*은 응용프로그램 서버(AS)나, 테이블 참조를 포함하는 fullselect 이전에 정의된 공통 테이블 표현식(483 페이지의 『공통 테이블 표현식』 참조)의 *테이블 이름*에서 기존의 테이블, 뷰 또는 별명을 식별해야 합니다. *테이블 이름*이 입력된 테이블을 참조하는 경우 이 이름은 모든 서브테이블과 함께 테이블의 UNION ALL을 나타내며, 이 때 *테이블 이름*의 컬럼만 표시됩니다. 마찬가지로, *뷰 이름*이 입력된 뷰를 참조하는 경우 이 이름은 모든 부속 뷰와 함께 뷰의 UNION ALL을 나타내며, 이 때 *뷰 이름*의 컬럼만 표시됩니다.

ONLY(*테이블 이름*) 또는 ONLY(*뷰 이름*)를 사용하는 것은 적절한 서브테이블 또는 부속 뷰의 행이 포함되지 않음을 의미합니다. ONLY와 함께 사용되는 *테이블 이름*에 서브테이블이 없는 경우, ONLY(*테이블 이름*)는 *테이블 이름*을 지정하는 것과 같습니다. ONLY와 함께 사용되는 *뷰 이름*에 부속 뷰가 없는 경우, ONLY(*뷰 이름*)는 *뷰 이름*을 지정하는 것과 같습니다.

OUTER(*테이블 이름*) 또는 OUTER(*뷰 이름*)를 사용하는 것은 가상 테이블을 나타냅니다. OUTER와 함께 사용되는 *테이블 이름*이나 *뷰 이름*에 서브테이블이나 부속 뷰가 없는 경우, OUTER를 지정하는 것은 OUTER를 지정하지 않는 것과 같습니다. OUTER(*table-name*)는 다음과 같이 *테이블 이름*에서 파생됩니다.

- 컬럼에는 *테이블 이름* 컬럼이 포함되며, 그 뒤에는 각 서브테이블에 의해 도입되는 추가 컬럼이 올 수 있습니다. 추가 컬럼은 오른쪽에 추가되므로 서브테이블 계층 순서가 depth-first 순서로 됩니다. 공통 상위를 가진 서브테이블은 유형 작성 순서대로 됩니다.
- 행에는 모든 *테이블 이름* 행들과 서브테이블 행들이 포함됩니다. 널(NULL) 값은 행에 대한 서브테이블에 없는 컬럼에 대해 리턴됩니다.

이전 포인트들은 OUTER(*뷰 이름*)에도 적용되어, *테이블 이름* 대신 *뷰 이름*으로, 서브테이블 대신 부속 뷰로 교체됩니다.

ONLY 또는 OUTER를 사용하려면 *table-name*의 모든 서브테이블이나 *view-name*의 모든 서브뷰에 SELECT 특권을 가져야 합니다.

테이블 참조로 지정되어 인수 유형과 함께 각 *함수 이름*은 응용프로그램 서버에서 기존의 테이블 함수로 환원되어야 합니다.

괄호로 묶인 fullselect는 (뒤의 상관 이름을) *중첩 테이블 표현식(nested table expression)*이라고 합니다.

*조인된 테이블*은 하나 이상의 조인 조작의 결과인 중간 결과 집합을 지정합니다. 445 페이지의 『조인된 테이블』에서 자세한 정보를 참조하십시오.

모든 테이블 참조의 표시 이름(exposed name)은 고유해야 합니다. 표시 이름은 다음과 같습니다.

- *상관 이름*,
- 뒤에 *상관 이름*이 오지 않는 *테이블 이름*,
- 뒤에 *상관 이름*이 오지 않는 *뷰 이름*,
- 뒤에 *상관 이름*이 오지 않는 *별명(nickname)*,
- 뒤에 *상관 이름*이 오지 않는 *별명(alias-name)*.

각 *상관 이름*은 바로 앞에 있는 *테이블 이름*, *뷰 이름*, *별명(nickname)*, *함수 이름* 참조 또는 중첩 테이블 표현식의 지정자로 정의됩니다. 테이블, 뷰, 테이블 함수 또는 중첩 테이블 표현식의 컬럼에 대해 규정화된 참조에는 표시 이름을 사용해야 합니다. 같은 테이블 이름, 뷰 또는 별명(nickname) 이름이 두 번 지정될 경우, 최소한 하나의 스펙 다음에 *상관 이름*이 와야 합니다. *상관 이름*은 테이블, 뷰

또는 별명의 컬럼에 대한 참조를 규정하기 위해 사용됩니다. *상관 이름*이 지정되면, *컬럼 이름*도 지정하여 *테이블 이름*, *뷰 이름*, *별명*,*함수 이름* 참조 또는 중첩 테이블 표현식의 컬럼에 이름을 지정할 수 있습니다. 149 페이지의 『상관 이름』에서 자세한 정보를 참조하십시오.

일반적으로, 테이블 함수와 중첩 테이블 표현식은 모든 from절에 지정할 수 있습니다. 테이블 함수와 중첩 테이블 표현식의 컬럼은 반드시 지정해야 하는 상관 이름을 사용하여 부속 선택의 나머지와 선택 목록에서 언급할 수 있습니다. 이 상관 이름의 범위는 FROM 절에서 다른 테이블, 뷰 또는 별명(nickname)에 대한 상관 이름과 같습니다. 다음의 경우 중첩 테이블 표현식을 사용할 수 있습니다.

- 뷰 작성을 피하기 위해 뷰 대신(뷰의 일반적인 사용이 필요하지 않을 경우)
- 원하는 결과 테이블이 호스트 변수를 기초로 할 경우

테이블 함수 참조

일반적으로, 테이블 함수와 같은 인수 값은 테이블이나 뷰에서와 똑같은 방법으로 SELECT의 FROM절에서 참조할 수 있습니다. 그러나, 몇가지 고려해야 할 사항이 있습니다.

- 테이블 함수 컬럼 이름

  상관 이름 다음에 대체 컬럼 이름이 제공되지 않는 경우, CREATE FUNCTION 문의 RETURN절에 지정되는 컬럼 이름이 테이블 함수에 대한 컬럼 이름입니다. 이것은 CREATE TABLE문에 정의되는 테이블의 컬럼 이름과 유사합니다. 테이블 함수 작성에 대해서는 685 페이지의 『CREATE FUNCTION(외부 테이블)』 또는 726 페이지의 『CREATE FUNCTION (SQL 스칼라, 테이블 또는 행)』에서 자세한 내용을 참조하십시오.

- 테이블 함수 차수

  함수 이름과 함께 테이블 함수 참조에 지정되는 인수는 *함수 차수*라는 알고리즘이 사용할 함수를 결정할 때 사용합니다. 이는 명령문에 사용되는 다른 함수(가령, 스칼라 함수)와 다르지 않습니다. 함수 차수에 대해서는 168 페이지의 『함수 차수』에서 설명됩니다.

- 테이블 함수 인수

  스칼라 함수 인수와 마찬가지로, 테이블 함수 인수도 유효한 임의의 SQL 표현식일 수 있습니다. 따라서, 다음은 유효한 구문입니다.

```
예 1:  SELECT c1
       FROM TABLE( tf1('Zachary') ) AS z
       WHERE c2 = 'FLORIDA';
예 2:  SELECT c1
       FROM TABLE( tf2 (:hostvar1, CURRENT DATE) ) AS z;
예 3:  SELECT c1
       FROM t
       WHERE c2 IN
               (SELECT c3  FROM
               TABLE( tf5 (t.c4) ) AS z  -- correlated reference
               )                          -- to previous FROM clause
```

테이블 참조에서의 상관 참조

상관 참조는 중첩 테이블 표현식 또는 테이블 함수에 인수로 사용될 수 있습니다. 이 두 경우에 적용되는 기본 규칙은 상관 참조가 부속 조회 계층의 하이 레벨 *테이블 참조*로부터 이어야 합니다. 이 계층에는 FROM의 왼쪽에서 오른쪽으로 처리에서 분석된 테이블 참조가 포함됩니다. 중첩 테이블 표현식의 경우, TABLE 키워드가 fullselect 앞에 나와야 합니다. 따라서, 다음은 유효한 구문입니다.

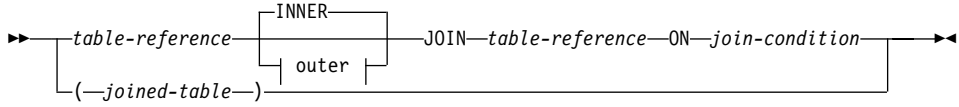
```
예 1:  SELECT t.c1, z.c5
       FROM t, TABLE( tf3(t.c2) ) AS z      -- t precedes tf3 in FROM
       WHERE t.c3 = z.c4;                   -- so t.c2 is known
예 2:  SELECT t.c1, z.c5
       FROM t, TABLE( tf4(2 * t.c2) ) AS z  -- t precedes tf3 in FROM
       WHERE t.c3 = z.c4;                   -- so t.c2 is known
예 3:  SELECT d.deptno, d.deptname
              empinfo.avgsal, empinfo.empcount
       FROM department d,
            TABLE (SELECT AVG(e.salary) AS avgsal,
                          COUNT(*) AS empcount
                   FROM employee e           -- department precedes and
                   WHERE e.workdept=d.deptno -- TABLE is specified
                  ) AS empinfo;              -- so d.deptno is known
```

그러나, 다음 예는 유효하지 않습니다.

```
예 4:  SELECT t.c1, z.c5
       FROM TABLE( tf6(t.c2) ) AS z, t     -- cannot resolve t in t.c2!
       WHERE t.c3 = z.c4;                  -- compare to Example 1 above.

예 5:  SELECT a.c1, b.c5
       FROM TABLE( tf7a(b.c2) ) AS a, TABLE( tf7b(a.c6) ) AS b
       WHERE a.c3 = b.c4;                   -- cannot resolve b in b.c2!
예 6:  SELECT d.deptno, d.deptname,
              empinfo.avgsal, empinfo.empcount
       FROM department d,
            (SELECT AVG(e.salary)AS avgsal,
                    COUNT(*) AS empcount
             FROM employee e           -- department precedes but
             WHERE e.workdept=d.deptno -- TABLE is not specified
            ) AS empinfo;              -- so d.deptno is unknown
```

조인된 테이블



outer:

*조인된 테이블(joined table)*은 내부 조인(inner join)이나 외부 조인(outer join)의 결과인 중간 결과 테이블을 지정합니다. 이 테이블은 조인 연산자, INNER, LEFT OUTER, RIGHT OUTER 또는 FULL OUTER 중 하나를 피연산자에 적용하여 얻어집니다.

내부 조인은 조인 조건이 참인 행만 유지하는 테이블의 벡터곱(왼쪽 테이블의 각 행을 오른쪽 테이블의 모든 행과 결합)으로 생각할 수 있습니다. 조인된 테이블의 하나 또는 전체에서 누락된 행이 결과 테이블이 될 수 있습니다. 외부 조인에는 내부 조인이 포함되며 이러한 누락 행이 보존됩니다. 세 가지 유형의 외부 조인이 있습니다.

1. ***왼쪽 외부 조인(left outer join)***에는 내부 조인으로부터 누락된 왼쪽 테이블의 행들이 포함됩니다.
2. ***오른쪽 외부 조인(right outer join)***에는 내부 조인으로부터 누락된 오른쪽 테이블의 행들이 포함됩니다.
3. ***완전 외부 조인(full outer join)***에는 내부 조인으로부터 누락된 오른쪽과 왼쪽 테이블의 행들이 포함됩니다.

조인 연산자(join-operator)가 지정되지 않은 경우, INNER가 사용됩니다. 여러 개의 조인시, 수행 순서는 결과에 영향을 미칩니다. 조인은 다른 조인 내에 중첩될 수 있습니다. 보통, 조인의 처리 순서는 왼쪽에서 오른쪽이지만, 요구되는 조인 조건의 위치에 따릅니다. 중첩된 조인의 경우, 괄호를 사용하여 순서를 명확히 하는 것이 좋습니다. 예를 들면,

```
tb1 left join  tb2 on tb1.c1=tb2.c1
    right join tb3 left join tb4 on tb3.c1=tb4.c1
                                    on tb1.c1=tb3.c1
```

아래와 같습니다.

```
(tb1 left join tb2 on tb1.c1=tb2.c1)
    right join (tb3 left join tb4 on tb3.c1=tb4.c1)
                                    on tb1.c1=tb3.c1
```

조인된 테이블은 SELECT문이 사용(형식에 관계없음)되는 어떤 문맥에서도 사용할 수 있습니다. 뷰나 커서의 SELECT문에 조인된 테이블이 포함되는 경우, 뷰나 커서는 읽기 전용입니다.

*조인 조건*은 다음을 제외하고 *검색 조건*입니다.

- 부속 조회, 스칼라 등을 포함할 수 없습니다.
- 참조 값이 오브젝트 식별자 컬럼이 아닌 참조해제(dereference) 연산 또는 DEREF 함수를 포함할 수 없습니다.
- 여기에는 SQL 함수가 포함될 수 없습니다.
- *조인 조건*의 표현식에 언급되는 컬럼은 (동일한 조인 테이블 절의 범위 내에서) 연관된 조인의 피연산자 테이블 중 하나의 컬럼이어야 합니다.
- 완전 외부 조인의 *조인 조건*에서 표현식에 언급되는 함수는 결정되어야 하며 외부 조치를 가지고 있으면 안 됩니다.

이러한 규칙(SQLSTATE 42972)에 따르지 않는 조인 조건인 경우 오류가 발생합니다.

컬럼 참조는 컬럼 이름 규정자의 분석 규칙을 통해 분석됩니다. 술어에 적용되는 규칙이 *조인 조건*에도 적용됩니다(216 페이지의 『술어』 참조).

조인 연산

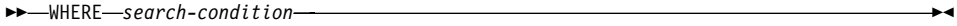
*조인 조건*은 한 쌍의 T1과 T2를 지정하며, 여기서 T1과 T2는 *조인 조건*의 JOIN 연산자의 왼쪽과 오른쪽 피연산자 테이블입니다. T1과 T2 행의 가능한 모든 조합에서, *조인 조건*이 참이면 T1의 행은 T2의 행과 짝을 이룹니다. T1의 행이 T2의 행과 조인되면, 결과의 행은 T2의 행의 값과 연결된 T1의 행의 값으로 구성됩니

다. 실행시 널(NULL) 행이 생성될 수 있습니다. 테이블의 널(NULL) 행은 컬럼에 널(NULL) 값이 허용되는지의 여부에 관계없이 테이블의 각 컬럼에 대한 널(NULL) 값으로 구성됩니다.

다음은 조인 조작의 결과를 요약한 것입니다.

- T1 INNER JOIN T2의 결과는 쌍을 이루는 이들 행으로 구성되며, 이 때 조인 조건은 참입니다.
- 조인 조건이 참인 때 T1 LEFT OUTER JOIN T2의 결과는 쌍을 이룬 이들 행 그리고 쌍을 이루지 않은 T1의 각 행과 T2의 널(NULL) 행의 병합으로 구성됩니다. T2에서 파생된 모든 컬럼에는 널(NULL) 값이 허용됩니다.
- 조인 조건이 참인 때 T1 RIGHT OUTER JOIN T2의 결과는 쌍을 이룬 이들 행 그리고 쌍을 이루지 않은 T2의 행과 T1의 널(NULL) 행의 병합으로 구성됩니다. T1에서 파생된 모든 컬럼에는 널(NULL) 값이 허용됩니다.
- T1 FULL OUTER JOIN T2의 결과는 쌍을 이룬 이들 행과, 쌍을 이루지 않은 T2의 각 행과 T1의 널(NULL) 행의 병합, 그리고 쌍을 이루지 않은 T1의 각 행과 T2의 널(NULL) 행의 병합으로 구성됩니다. T1과 T2에서 파생된 모든 컬럼에는 널(NULL) 값이 허용됩니다.

where절



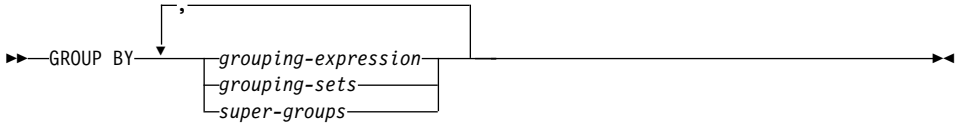
WHERE 절은 *검색 조건*이 참인 R의 행들로 구성되는 중간 결과 테이블을 지정합니다. R은 부속 선택의 FROM절에 대한 결과입니다.

*검색 조건*은 다음 규칙에 따라야 합니다.

- *각 컬럼 이름*은 R의 컬럼을 명백히 식별하거나 상관 참조여야 합니다. *컬럼 이름*은 외부 부속 선택에서 *테이블 참조*나 뷰의 컬럼을 식별할 경우에 상관 참조입니다.
- 컬럼 함수는 WHERE절이 Having절의 부속 조회에 지정되고 함수의 인수가 그룹에 대한 상관 참조가 아니면 지정할 수 없습니다.

*검색 조건*에서의 부속 조회는 R의 각 행에 대해 효과적으로 실행되고, 그 결과는 R의 주어진 행에 대한 *검색 조건*의 응용프로그램에서 사용됩니다. 부속 조회는 실제로 상관 참조를 포함하고 있는 경우에만 R의 각 행에 대해 실행됩니다. 사실상, 상관 참조가 없는 부속 조회는 한번 실행되는데 비해, 상관 참조가 있는 부속 조회는 각 행에 대해 한번씩 실행되어야 할 수도 있습니다.

group-by절



GROUP BY절은 R의 행의 그룹화로 이루어지는 중간 결과 테이블을 지정합니다. R은 부속 선택에 대한 이전 절의 결과입니다.

가장 간단한 형식으로, GROUP BY절에는 *그룹 표현식*이 포함됩니다. 그룹 표현식은 R 그룹을 정의할 때 사용되는 *표현식*입니다. 그룹 표현식에 포함된 각 *컬럼 이름*은 R의 컬럼을 확실하게 식별해야 합니다(SQLSTATE 42702 또는 42703). 각 그룹 표현식의 길이 속성은 255 바이트보다 클 수 없습니다(SQLSTATE 42907). 그룹 표현식에는 스칼라 fullselect(SQLSTATE 42822)나, 유형이 다르거나 외부 조치가 있는 함수(SQLSTATE 42845)는 포함할 수 없습니다.

더 복잡한 GROUP BY절에는 *그룹 세트*와 *수퍼 그룹*이 포함됩니다. 이 형식의 자세한 설명에 대해서는 450 페이지의 『그룹 세트』와 451 페이지의 『수퍼 그룹』을 각각 참조하십시오.

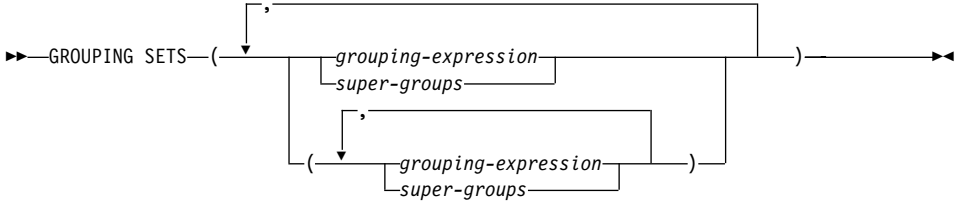
GROUP BY의 결과는 행 그룹의 집합입니다. 결과의 각 행은 *그룹 표현식*이 동일한 행 집합을 나타냅니다. 그룹화의 경우, *그룹 표현식*의 모든 널(NULL) 값은 동일하게 간주됩니다.

*그룹 표현식*은 HAVING절의 검색 조건에, SELECT절의 표현식에 또는 ORDER BY절의 *sort-key-표현식*에서 사용할 수 있습니다(자세한 사항은 486 페이지의 『order-by절』 참조). 각 경우에서, 참조사항은 각 그룹에 대해 하나의 값만 지정합니다. 예를 들어, *그룹 표현식*이 *col1+col2*이면, 선택 목록에서 허용되는 표현식은 *col1+col2+3*입니다. 표현식에 대한 연관성 규칙에 의해, 유사한 표현식인 *3+col1+col2*는 괄호를 사용하여 해당되는 표현식이 동일한 순서로 평가되도록 하지 않으면 허용되지 않습니다. 따라서, *3+(col1+col2)*도 선택 목록에서 사용할 수 있습니다. 병합 연산자가 사용되는 경우, *그룹 표현식*은 선택 목록에 지정된 표현식에서와 동일하게 사용되어야 합니다.

*그룹 표현식*에 후미 공백이 있는 가변 길이의 문자열이 포함된 경우, 그룹의 값은 후미 공백의 수에서 다를 수 있으므로 길이가 다를 수 있습니다. 이 경우, *그룹 표현식*을 언급하면 각 그룹에 대해 하나의 값만 지정되지만, 그룹에 대한 값은 사용 가능한 값 집합에서 임의대로 선택됩니다. 그러므로, 결과 값의 실제 길이는 예측할 수 없습니다.

GROUP BY절이 SELECT절에 지정된 컬럼을 표현식(스칼라 fullselect, 변형 함수 또는 외부 조치 함수)으로 직접 언급할 수 없는 경우가 몇 가지 있습니다. 그러한 표현식을 사용하여 그룹화하려면, 중첩 테이블 표현식이나 공통 테이블 표현식을 사용하여 먼저 결과 테이블에 결과의 컬럼으로 표현식을 제공하십시오. 중첩 테이블 표현식을 사용하는 예에 대해서는 461 페이지의 예 A9에서 참조하십시오.

그룹 세트



그룹 세트 스펙은 단일 명령문에 지정된 여러 그룹 절을 허용합니다. 이것은 두 개 이상의 행의 그룹을 단일 결과 집합으로의 통합으로 여겨집니다. 이는 한 개의 그룹 세트에 해당하는 각 부속 선택의 절로써 그룹이 있는 여러 부속 선택의 통합과 논리적으로 동일합니다. 그룹 세트는 단일 요소가 될 수 있거나 괄호로 한계가 정해진 요소의 목록이 될 수 있으며 여기서 요소는 그룹 표현식 이거나 수퍼 그룹이 됩니다. *그룹 세트*의 사용에서는 그룹이 계산되어지는 것을 허용합니다.

그룹 세트 스펙에서는 간단한 *그룹 표현식*이 사용되거나 더 복잡한 *수퍼 그룹*의 형식을 허용합니다. *수퍼 그룹*의 451 페이지의 『수퍼 그룹』에서 자세한 설명을 참조하십시오.

그룹 세트는 GROUP BY 연산에 대한 기본적인 내장 블록인 것에 유의하십시오. 단일 컬럼으로 된 단순한 그룹은 한 개의 요소로 된 그룹 세트로 간주될 수 있습니다. 예를 들면,

GROUP BY는

아래와 같습니다.

```
GROUP BY GROUPING SET((a))
```

와

```
GROUP BY a,b,c는
```

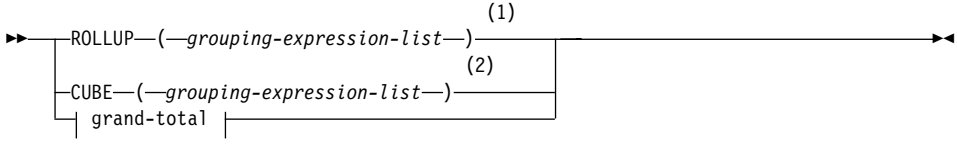
아래와 같습니다.

```
GROUP BY GROUPING SET((a,b,c))
```

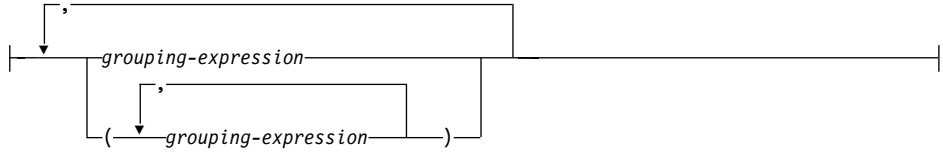
그룹 세트에서 제외된 부속 선택의 선택 목록의 비총계 컬럼은 그 그룹 세트에 대해 생성되어진 각행 컬럼의 널(NULL)을 리턴할 것입니다. 이것은 컬럼 값의 총계가 이 컬럼에 대한 값을 고려하지 않았다는 사실에 영향을 미칩니다. 행에서 실제 데이터의 널(NULL)과 그룹 세트에서 생성된 널(NULL)이 있는 행의 구별 방법은 268 페이지의 『GROUPING』에서 참조하십시오.

467 페이지의 Example C2에서 472 페이지의 예 C7은 그룹 세트의 쓰임을 보여줍니다.

수퍼 그룹



grouping-expression-list:



grand-total:

주:

1 group-by절에서 단독으로 사용될 경우의 대체 스펙: 그룹 표현식 목록 WITH ROLLUP.

2 group-by절에서 단독으로 사용될 경우의 대체 스펙: 그룹 표현식 목록 WITH CUBE

ROLLUP (*그룹 표현식 목록*)

*ROLLUP 그룹화*는 정규의 그룹 행외에 *소계*행을 포함하는 결과 집합을 생성하는 GROUP BY절의 연장입니다. *소계* 행[53]그룹 행을 확보하기 위해 사용된 것과 동일한 컬럼 함수를 적용하여 그 값이 얻어지는 추가 합산을 포함하는 "추가 총계" 행입니다.

ROLLUP 그룹화는 일련의 *그룹 세트*입니다. *n*개의 요소로 된 ROLLUP의 일반적 스펙은

```
GROUP BY ROLLUP(C1,C2,...,Cn-1,Cn)
```

아래와 동일합니다.

```
GROUP BY GROUPING SETS((C1,C2,...,Cn-1,Cn)
                       (C1,C2,...,Cn-1)
                       ...
                       (C1,C2)
                       (C1)
                       () )
```

ROLLUP의 *n*개의 요소가 *n*+1 그룹 세트로 바뀌는 것에 유의하십시오.

ROLLUP의 경우 *그룹 표현식*의 지정 순서는 매우 중요합니다. 예를 들면,

```
GROUP BY ROLLUP(a,b)
```

는

아래와 동일합니다

```
GROUP BY GROUPING SETS((a,b)
                       (a)
                       () )
```

53. 이것이 가장 일반적인 쓰임이므로 소계 행이라고 하지만, 컬럼 함수는 모두 총계에 사용할 수 있습니다. 예를 들어, MAX 및 AVG가 474 페이지의 예 C8에서 사용됩니다.

반면

GROUP BY ROLLUP(b,a**)**는

아래와 같습니다.

```
GROUP BY GROUPING SETS((b,a)
                       (b)
                       () )
```

ORDER BY절은 결과 집합의 행 순서를 보장하는 유일한 방법입니다. 467 페이지의 예 C3은 ROLLUP의 쓰임을 보여줍니다.

CUBE (*그룹 표현식 목록*)

*CUBE 그룹화*는 ROLLUP 총계의 모든 행과 이에 더하여 ″교차 테이블″ 행도 포함하는 결과 집합을 산출하는 GROUP BY절의 연장입니다. *교차 테이블* 행은 소계가 있는 총계의 일부가 아닌 부가적인 ″추가 총계″ 행입니다.

ROLLUP처럼, CUBE 그룹화는 일련의 *그룹 세트*로 여겨질 수 있습니다. CUBE 경우, 큐브된 *그룹 표현식* 목록의 모든 순열은 총계에 따라 계산됩니다. 그러므로, CUBE의 *n*개의 요소는 2**n(2 힘 *n*) *그룹 세트*로 바뀝니다. 예를 들어,

GROUP BY CUBE(a,b,c**)**의 스펙은

아래와 동일합니다

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (b,c)
                       (a)
                       (b)
                       (c)
                       () )
```

CUBE의 3개 요소가 8개 그룹 세트로 바뀐것에 유의하십시오.

CUBE의 경우 요소의 스펙 순서는 중요하지 않습니다. CUBE(DayOfYear, Sales_Person)'와 'CUBE (Sales_Person, DayOfYear)'는 동일한 결과 집합을 산출합니다. '동일한'이라는 단어는 결과 집합의 내용에 대한 것이지 이의

순서에 대한 것은 아닙니다. ORDER BY절은 결과 집합의 행 순서를 보장하는 유일한 방법입니다. 468 페이지의 예 C4는 CUBE의 쓰임을 보여줍니다.

그룹 표현식 목록

*그룹 표현식 목록*은 CUBE 내에 사용되거나 CUBE 요소의 번호를 정의하는 ROLLUP절 또는 ROLLUP 연산에 사용됩니다. 이는 여러 *그룹 표현식*으로 된 요소의 한계를 정하는 괄호를 사용하여 제어할 수 있습니다.

*그룹 표현식*에 대한 규칙은 449 페이지의 『group-by절』에서 설명됩니다. 예를 들어, 조회가 Province 내에 있는(County 내는 제외) City의 ROLLUP에 대한 총 비용을 리턴한다고 가정합시다. 그러나, 다음 절은

```
GROUP BY ROLLUP (Province, County, City)
```

County에 대해 원하지 않는 소계를 산출합니다. 아래 절에서

```
GROUP BY ROLLUP(Province, (County, City))
```

복합 구성요소(County, City)는 ROLLUP에서 하나의 구성요소를 이루므로, 이 절을 사용하는 조회는 원하는 결과를 가져옵니다. 다시 말해서, 두 개의 요소 ROLLUP

```
GROUP BY ROLLUP(Province, (County, City))
```

생성합니다

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province)
                       () )
```

세 개의 요소 ROLLUP이 생성되는 동안

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province, County)
                       (Province)
                       () )
```

467 페이지의 Example C2에서도 복합 컬럼 값을 활용하고 있습니다.

총계

CUBE와 ROLLUP은 모두 전체적인 총계 행을 리턴합니다. 이는 GROUPING SET절 내에서 공백 괄호를 사용하여 별도로 지정할 수 있습니다. 조회의 결

과에 영향을 주지 않으면서 GROUP BY절 내에 직접 지정할 수 있습니다. 468 페이지의 예 C4에서는 총계 구문을 사용합니다.

그룹 세트 조합

이는 GROUP BY절의 임의의 유형을 결합하는 데 사용할 수 있습니다. 간단한 *그룹 표현식* 필드는 다른 그룹과 결합하는 경우, 결과 *그룹 세트*의 시작에 추가됩니다. ROLLUP 또는 CUBE 표현식들과 조합될 때, 이들은 ROLLUP이나 CUB의 정의에 따라서 추가적 그룹 세트 등록을 형성하면서, 기존의 표현식에 ″multipliers″처럼 작용합니다.

예를 들어, *그룹 표현식* 요소를 결합하면 다음과 같이 작동합니다

```
GROUP BY a, ROLLUP(b,c)는
```

아래와 동일합니다

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a) )
```

또는 유사하게,

```
GROUP BY a, b, ROLLUP(c,d)는
```

아래와 동일합니다

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b) )
```

ROLLUP 요소를 결합하면 다음과 같이 작동합니다.

```
GROUP BY ROLLUP(a), ROLLUP(b,c)는
```

아래와 동일합니다

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a)
                       (b,c)
                       (b)
                       () )
```

유사하게,

GROUP BY ROLLUP(a), **CUBE**(b,c)는

아래와 동일합니다

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (a)
                       (b,c)
                       (b)
                       (c)
                       () )
```

*CUBE*와 *ROLLUP* 요소를 결합하면 다음과 같이 작동합니다.

GROUP BY CUBE(a,b), **ROLLUP**(c,d)는

아래와 동일합니다

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b)
                       (a,c,d)
                       (a,c)
                       (a)
                       (b,c,d)
                       (b,c)
                       (b)
                       (c,d)
                       (c)
                       () )
```

간단한 *그룹 표현식*처럼, 그룹 세트의 조합은 또한 각 그룹 세트 내의 중복을 제거합니다. 예를 들어,

GROUP BY a, **ROLLUP**(a,b)는

아래와 동일합니다

```
GROUP BY GROUPING SETS((a,b)
                       (a) )
```

더 복잡한 그룹 세트 조합의 예가 완전 CUBE 총계에 대해 리턴될 특정 행을 제거하는 결과 집합을 구성하는 것입니다.

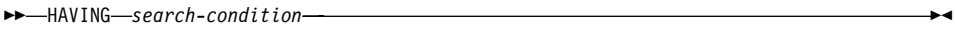
예를 들어, 다음 GROUP BY절을 생각해봅시다.

```
GROUP BY Region,
         ROLLUP(Sales_Person, WEEK(Sales_Date)),
         CUBE(YEAR(Sales_Date), MONTH (Sales_Date))
```

GROUP BY 바로 오른쪽에 나열되는 컬럼은 단순하게 그룹화되며, ROLLUP 다음 괄호 내의 컬럼은 롤업되고 CUBE 다음 괄호 내의 컬럼은 큐브로 만들어집니다. 따라서, 위의 절은 YEAR 안의 MONTH가 큐브로 되며, 이는 다시 Region 총계내의 Sales_Person 내의 WEEK 안에서 롤업됩니다. Region, Sales_Person 또는 WEEK(Sales_Date)에 대해 총계 행이나 교체 테이블 행을 생성하지 않으므로, 아래 절보다 생성하는 행 수는 더 작습니다.

```
GROUP BY ROLLUP (Region, Sales_Person, WEEK(Sales_Date),
                 YEAR(Sales_Date), MONTH(Sales_Date) )
```

having절



HAVING 절은 *검색 조건*이 참인 R의 그룹들로 구성되는 중간 결과 테이블을 지정합니다. R은 부속 선택에 대한 이전 절의 결과입니다. 이 절이 GROUP BY가 아니면, R은 그룹화 컬럼이 없는 단일 그룹으로 간주됩니다.

검색 조건에서 각 *컬럼 이름*은 다음 중 하나여야 합니다.

- R의 그룹화 컬럼을 명백히 식별합니다.
- 컬럼 함수 내에 지정해야 합니다.
- 상관 참조여야 합니다. *컬럼 이름*은 외부 부속 선택에서 *테이블 참조*나 뷰의 컬럼을 식별할 경우에 상관 참조입니다.

검색 조건이 적용되는 R의 그룹은 인수가 상관 참조인 함수를 제외하고, 검색 조건의 각 컬럼 함수에 대해 인수를 제공합니다.

검색 조건에 부속 조회가 있으면, 검색 조건이 R 그룹에 적용될 때마다, 그리고 검색 조건 적용시 그 결과를 사용할 때마다 부속 조회가 실행되는 것으로 생략될

수 있습니다. 실제로, 상관 참조가 포함되어 있을 경우에만 각 그룹에 대해 부속 조회가 실행됩니다. 차이에 대해서는 460 페이지의 예 A6과 460 페이지의 예 A7에서 참조하십시오.

R의 그룹에 대한 상관 참조는 그룹화 컬럼을 식별하거나 컬럼 함수 내에 포함되어야 합니다.

HAVING이 GROUP BY없이 사용되면, 선택 목록은 컬럼 함수, 상관 컬럼 참조, 리터럴 또는 특수 레지스터 내의 컬럼 이름만 될 수 있습니다.

부속 선택의 예

예 A1: EMPLOYEE 테이블에서 모든 컬럼과 행을 선택합니다.

```
SELECT * FROM EMPLOYEE
```

예 A2: EMP_ACT와 EMPLOYEE 테이블을 조인하고, EMP_ACT 테이블에서 모든 컬럼을 선택한 후 사원의 성(LASTNAME)을 EMPLOYEE 테이블에서 결과의 각 행에 추가합니다.

```
SELECT EMP_ACT.*, LASTNAME
 FROM EMP_ACT, EMPLOYEE
 WHERE EMP_ACT.EMPNO = EMPLOYEE.EMPNO
```

예 A3: EMPLOYEE와 DEPARTMENT 테이블을 조인하고, 사원 번호(EMPNO), 사원 성(LASTNAME), 부서 번호(EMPLOYEE 테이블의 WORKDEPT와 DEPARTMENT 테이블의 DEPTNO), 그리고 1930년 이전에 생긴(BIRTHDATE) 부서 이름(DEPTNAME)을 선택합니다.

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
   FROM EMPLOYEE, DEPARTMENT
   WHERE WORKDEPT = DEPTNO
   AND YEAR(BIRTHDATE) < 1930
```

예 A4: EMPLOYEE 테이블에서 동일한 작업 코드를 갖는 각 행 그룹과 최대 급여가 27000 이상이고 하나 이상의 행을 갖는 그룹에 대해서만 작업(JOB)과 최대 및 최소 급여(SALARY)를 선택합니다.

```
SELECT JOB, MIN(SALARY),  MAX(SALARY)
    FROM EMPLOYEE
   GROUP BY JOB
   HAVING COUNT(*) > 1
   AND MAX(SALARY) >= 27000
```

예 A5: 부서(WORKDEPT) 'E11'에서 사원(EMPNO)에 대한 EMP_ACT 테이블의 모든 행을 선택합니다(사원 부서 번호는 EMPLOYEE 테이블에 표시됩니다.).

```
   SELECT *
    FROM EMP_ACT
    WHERE EMPNO IN
```

```
        (SELECT EMPNO
            FROM EMPLOYEE
            WHERE WORKDEPT = 'E11' )
```

예 A6: EMPLOYEE 테이블을 사용하여, 최대 급여가 전체 부서의 평균 급여보다 작은 모든 부서에 대해 부서 번호(WORKDEPT)와 부서의 최대 급여(SALARY)를 선택합니다.

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG( SALARY)
                        FROM EMPLOYEE )
```

HAVING절에서의 부속 조회는 이 예에서 한번만 실행됩니다.

예 A7: EMPLOYEE 테이블을 사용하여, 최대 급여가 다른 모든 부서의 평균 급여보다 작은 모든 부서에 대해 부서 번호(WORKDEPT)와 부서의 최대 급여(SALARY)를 선택합니다.

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE EMP_COR
   GROUP BY WORKDEPT
   HAVING MAX(SALARY) < (SELECT AVG( SALARY)
                         FROM EMPLOYEE
                         WHERE NOT WORKDEPT = EMP_COR.WORKDEPT)
```

예 A6에서와는 달리, HAVING절 내의 부속 조회는 각 그룹에 대해 실행되어야 합니다

예 8: 부서의 평균 급여 및 사원 수와 함께, 영업 대표들의 급여와 사원 수를 판별합니다.

이 조회는 먼저 중첩 테이블 표현식(DINFO)을 작성하여, AVGSALARY와 EMPCOUNT 컬럼, WHERE절에서 사용되는 DEPTNO 컬럼을 확보해야 합니다.

```
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY, DINFO.AVGSALARY, DINFO.EMPCOUNT
 FROM EMPLOYEE THIS_EMP,
       (SELECT OTHERS.WORKDEPT AS  DEPTNO,
                        AVG(OTHERS.SALARY ) AS AVGSALARY,
                        COUNT(*) AS EMPCOUNT
         FROM EMPLOYEE OTHERS
         GROUP BY OTHERS.WORKDEPT
```

```
      ) AS DINFO
WHERE THIS_EMP.JOB = 'SALESREP'
 AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

이 경우에 중첩 테이블 표현식을 사용하면, 정규 뷰로 DINFO 뷰를 작성할 때의 오버헤드가 감소됩니다. 명령문 준비시, 영업 대표들의 부서에 해당되는 행들인 조회의 나머지 문맥이 뷰에서 고려되어야 하므로, 뷰에 대한 카탈로그 액세스를 피할 수 있습니다.

예 A9: 5개의 무작위 사원 그룹에 대한 평균 교육 수준과 급여를 표시합니다.

이 조회에서는 GROUP BY절에 후속하여 사용될 수 있도록, 중첩 테이블 표현식을 사용하여 각 사원에 대한 무작위 값을 설정해야 합니다.

```
SELECT RANDID , AVG( EDLEVEL), AVG(SALARY )
  FROM ( SELECT EDLEVEL, SALARY, INTEGER(RAND()*5) AS RANDID
           FROM EMPLOYEE
       ) AS EMPRAND
  GROUP BY RANDID
```

조인의 예

예 B1: 이 예는 J1과 J2를 사용하여 다양한 조인의 결과를 보여줍니다. 이러한 테이블들은 나타난 대로 행을 포함합니다.

```
SELECT * FROM J1
W   X
--- ------
A       11
B       12
C       13
SELECT * FROM J2
Y   Z
--- ------
A       21
C       22
D       23
```

다음 조회는 두 테이블의 첫번째 컬럼과 일치하는 J1과 J2의 내부 조인입니다.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y
W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
```

이 내부 조인 예의 결과에는 J1의 W='C' 컬럼을 갖는 행과 J2의 Y='D' 컬럼을 갖는 행을 포함하고 있지 않습니다. 왜냐하면 이들은 다른 테이블에서는 일치하고 있지 않기 때문입니다. 다음의 내부 조인 조회의 변경된 형태는 이러한 결과를 제시해줍니다.

```
SELECT * FROM J1, J2 WHERE W=Y
```

다음 왼쪽 외부 조인은 J2 컬럼에 대해 널(NULL) 값을 가지면서, J1의 사라진 행을 찾게 합니다. 즉, 모든 J1 행을 포함하고 있습니다.

```
SELECT * FROM J1 LEFT OUTER JOIN J2 ON W=Y
W   X      Y   Z
--- ------ --- ------
A       11 A       21
B       12 -        -
C       13 C       22
```

다음 오른쪽 외부 조인은 J1 컬럼에 대해 널(NULL) 값을 가지면서, J2의 사라진 행을 찾게 합니다. 즉, 모든 J2 행을 포함하고 있습니다.

```
SELECT * FROM J1 RIGHT OUTER JOIN J2 ON W=Y
W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
```

다음 완전 외부 조인은 적절한 곳에 널(NULL) 값을 가지면서 J1과 J2의 사라진 행을 찾게 합니다. 즉, J1과 J2의 모든 행은 포함하고 있습니다.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
B       12 -        -
```

예 B2: 이전 예에서 테이블 J1과 J2을 사용하여, 추가된 술어가 검색 조건에 첨부될 때 언제, 어떤일이 일어나는지 관찰하십시오.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=13
W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

추가 조건으로 내부 조인은 462 페이지의 예 B1에 있는 내부 조인과 비교하여 한 개의 행만 선택합니다

이러한 영향은 완전 외부 조인에 있다는 것에 유의하십시오.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=13
W   X      Y   Z
--- ------ --- ------
-        - A       21
C       13 C       22
-        - D       23
A       11 -        -
B       12 -        -
```

이제, 결과는 5개의 행(추가 술어가 없을땐 4개)을 갖고 있습니다. 이는 내부 조인시 단 한개의 행이 있었고 양쪽 테이블의 모든 행이 나타났기 때문입니다.

다음에 오는 조회는 WHERE절에 동일한 추가 술어를 사용한 것이 완전히 다른 결과를 야기시킨다는 것을 보여줍니다.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=13
W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

WHERE절은 완전 외부 조인의 중간 결과 후에 적용되야 합니다. 이 중간 결과는 예 B1 462 페이지의 예 B1에 나오는 완전 외부 조인 조회 결과와 유사할 것입니다. WHERE절은 중간 결과에 적용되고 X=13을 갖는 행을 제외하고 모두 제거됩니다. 따라서 외부 조인을 수행할 때 술어의 위치 선택은 결과에 상당한 영향을 줄 수 있습니다. 술어가 X=13 대신에 X=12이 된다면 어떻게 될지 고려해 보십시오. 다음 내부 조인은 아무 행도 나타나지 않습니다.

```
   SELECT * FROM J1
INNER JOIN J2 ON
W=Y AND X=12
```

그러므로, 완전 외부 조인은 6개의 행(J2의 컬럼에 대해 널(NULL)을 갖는 J1에서 세 행, 그리고 J1 컬럼에 대해 널(NULL)을 갖는 J2에서 세 행)을 리턴합니다.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=12
W   X      Y   Z
--- ------ --- ------
-         - A       21
-         - C       22
-         - D       23
A       11 -         -
B       12 -         -
C       13 -         -
```

이 추가 술어가 WHERE절에 쓰인다면, 한개의 행만이 표시됩니다.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=12
W   X      Y   Z
--- ------ --- ------
B       12 -         -
```

예 B3: 사원 번호와 관리자의 성을 갖는 모든 부서를 관리자가 없는 부서들과 함께 나열하십시오.

```
SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
  FROM DEPARTMENT LEFT OUTER JOIN EMPLOYEE
       ON MGRNO = EMPNO
```

예 B4: 모든 사원 번호와 그 번호를 갖는 사람의 성 그리고 그들의 관리자의 성들을 관리자가 없는 사원들과 함께 나열하십시오.

```
SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME
  FROM EMPLOYEE E LEFT OUTER JOIN
                                        DEPARTMENT INNER JOIN EMPLOYEE M
       ON MGRNO = M.EMPNO
       ON E.WORKDEPT = DEPTNO
```

내부 조인은 DEPARTMENT 테이블에 확인된 관리자에 대한 성을 규정하고, 왼쪽 외부 조인은 해당 하는 부서가 DEPARTMENT에 없더라도 각 사원이 나열되도록 보장합니다.

그룹 세트, Cube와 Rollup의 예

Example C1에서 468 페이지의 예 C4 까지의 조회들은 'WEEK(SALES_DATE) = 13'이란 술어를 기반으로 하는 SALES 테이블 행들의 부속 집합을 사용하고 있습니다.

```
SELECT WEEK(SALES_DATE) AS WEEK,
                          DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SALES AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
```

결과:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 LUCCHESSI                 3
         13           6 LUCCHESSI                 1
         13           6 LEE                       2
         13           6 LEE                       2
         13           6 LEE                       3
         13           6 LEE                       5
         13           6 GOUNOT                    3
         13           6 GOUNOT                    1
         13           6 GOUNOT                    7
         13           7 LUCCHESSI                 1
         13           7 LUCCHESSI                 2
         13           7 LUCCHESSI                 1
         13           7 LEE                       7
         13           7 LEE                       3
         13           7 LEE                       7
         13           7 LEE                       4
         13           7 GOUNOT                    2
         13           7 GOUNOT                   18
         13           7 GOUNOT                    1
```

예 C1: 다음은 기본 GROUP BY 절이 세 컬럼을 넘는 조회입니다.

```
SELECT WEEK(SALES_DATE) AS WEEK,
                          DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
                          SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
```

```
WHERE WEEK(SALES_DATE) = 13
GROUP BY WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

결과:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
```

예 C2: SALES 테이블 행들의 두 개의 다른 그룹 세트에 근거하는 결과를 제시하고 있습니다.

```
SELECT WEEK(SALES_DATE) AS WEEK,
                           DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
                           SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY GROUPING SETS ( (WEEK(SALES_DATE), SALES_PERSON),
                         (DAYOFWEEK(SALES_DATE), SALES_PERSON))
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

결과:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           - GOUNOT                   32
         13           - LEE                      33
         13           - LUCCHESSI                 8
          -           6 GOUNOT                   11
          -           6 LEE                      12
          -           6 LUCCHESSI                 4
          -           7 GOUNOT                   21
          -           7 LEE                      21
          -           7 LUCCHESSI                 4
```

WEEK 13을 갖는 행들은 첫번째 그룹 세트에서, 그 밖의 다른 행들은 두 번째 그룹 세트에서 각각 기인된 것입니다.

예 C3: 만일 예 C2 467 페이지의 Example C2의 그룹 세트에 포함되어 있는 3 개의 변별적 컬럼들을 사용하면서, ROLLUP을 수행한다면 (WEEK,DAY_WEEK, SALES_PERSON), (WEEK, DAY_WEEK), (WEEK) 그리고 총계에 대한 그룹 세트들을 볼 수 있습니다

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
  FROM SALES
  WHERE WEEK(SALES_DATE) = 13
  GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
  ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

결과:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           6 -                        27
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
         13           7 -                        46
         13           - -                        73
          -           - -                        73
```

예 C4: ROLLUP 대신 CUBE만을 대체하고, 예 C2 467 페이지의 예 C3과 동일한 조회를 실행한다면, 그 결과에, (WEEK,SALES_PERSON), (DAY_WEEK,SALES_PERSON), (DAY_WEEK), (SALES_PERSON)에 대한 추가 그룹 집합을 볼 수 있습니다

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY CUBE ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

결과:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
```

```
13          6 -                    27
13          7 GOUNOT               21
13          7 LEE                  21
13          7 LUCCHESSI             4
13          7 -                    46
13          - GOUNOT               32
13          - LEE                  33
13          - LUCCHESSI             8
13          - -                    73
 -          6 GOUNOT               11
 -          6 LEE                  12
 -          6 LUCCHESSI             4
 -          6 -                    27
 -          7 GOUNOT               21
 -          7 LEE                  21
 -          7 LUCCHESSI             4
 -          7 -                    46
 -          - GOUNOT               32
 -          - LEE                  33
 -          - LUCCHESSI             8
 -          - -                    73
```

예 C5: SALES_PERSON 과 MONTH을 갖는 그룹 행과 함께, SALES 테이블에서 선택한 행들의 총계를 나타내는 결과 집합을 구하십시오.

```
SELECT SALES_PERSON,
       MONTH(SALES_DATE) AS MONTH,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS ( (SALES_PERSON, MONTH(SALES_DATE)),
                         ()
                       )
ORDER BY SALES_PERSON, MONTH
```

결과:

```
SALES_PERSON    MONTH       UNITS_SOLD
--------------- ----------- -----------
GOUNOT                    3          35
GOUNOT                    4          14
GOUNOT                   12           1
LEE                       3          60
LEE                       4          25
LEE                      12           6
LUCCHESSI                 3           9
```

```
LUCCHESSI                4           4
LUCCHESSI               12           1
          - -                       155
```

예 C6: 이 예는 단일 결과 집합에서 그룹 집합으로써 두 ROLLUP을 다루면서 그룹 집합에 포함된 각 컬럼에 대해 순화적인 행을 지정하는 한 개의 조회가 수반된 2개의 단순 ROLLUP 조회를 보여줍니다.

예 C6-1:

```
SELECT WEEK(SALES_DATE) AS WEEK,
                           DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) )
ORDER BY WEEK, DAY_WEEK
```

결과:

```
WEEK        DAY_WEEK    UNITS_SOLD
----------- ----------- -----------
         13           6          27
         13           7          46
         13           -          73
         14           1          31
         14           2          43
         14           -          74
         53           1           8
         53           -           8
          - -                      155
```

예 C6-2:

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( MONTH(SALES_DATE), REGION );
ORDER BY MONTH, REGION
```

결과:

```
MONTH       REGION          UNITS_SOLD
----------- --------------- -----------
          3 Manitoba                 22
          3 Ontario-North             8
          3 Ontario-South            34
```

```
 3 Quebec                  40
 3 -                      104
 4 Manitoba                17
 4 Ontario-North            1
 4 Ontario-South           14
 4 Quebec                  11
 4 -                       43
12 Manitoba                 2
12 Ontario-South            4
12 Quebec                   2
12 -                        8
 - -                      155
```

예 C6-3:

```
SELECT WEEK(SALES_DATE) AS WEEK,
                        DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS ( ROLLUP( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) ),
                         ROLLUP( MONTH(SALES_DATE), REGION )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

결과:

```
WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
----------- ----------- ----------- --------------- -----------
         13           6           - -                        27
         13           7           - -                        46
         13           -           - -                        73
         14           1           - -                        31
         14           2           - -                        43
         14           -           - -                        74
         53           1           - -                         8
         53           -           - -                         8
          -           -           3 Manitoba                 22
          -           -           3 Ontario-North             8
          -           -           3 Ontario-South            34
          -           -           3 Quebec                   40
          -           -           3 -                       104
          -           -           4 Manitoba                 17
          -           -           4 Ontario-North             1
          -           -           4 Ontario-South            14
          -           -           4 Quebec                   11
          -           -           4 -                        43
          -           -          12 Manitoba                  2
          -           -          12 Ontario-South             4
          -           -          12 Quebec                    2
```

```
-           -           12 -                    8
-           -            - -                  155
-           -            - -                  155
```

그룹 집합으로써 두 ROLLUP을 사용하는 것은 중복행을 갖는 결과 집합을 유도합니다. 또 심지어 2개의 총계 행을 나타냅니다.

ORDER BY의 사용이 결과에 어떤 영향을 미쳤는지 주목하십시오.

- 첫번째 그룹 집합에서, 53주의 위치가 끝으로 이동되었습니다.
- 두 번째 그룹 집합에서, 12월의 위치가 끝으로 이동되고, 지역은 알파벳순으로 표시됩니다.
- 널(NULL) 값은 맨 위에 정렬됩니다.

예 C7: 단일 패스에서 여러 개의 ROLLUP을 수행하는 조회에서(471 페이지의 예 C6-3와 같은) 어느 그룹화 집합이 행을 만들어 냈는지를 나타내고자 할 수도 있습니다.

단계 1: VALUES 절(fullselect의 대체 양식)에서 선택하는 조회를 사용하여 새로운 데이터 값을 "생성"하는 방법을 소개합니다.

```
SELECT R1,R2
FROM (VALUES('GROUP 1','GROUP 2')) AS X(R1,R2);
```

결과:

```
R1      R2
------- -------
GROUP 1 GROUP 2
```

단계 2: SALES 테이블과 함께 이 테이블 "X"의 상호 제품을 형성합니다. 이것은 모든 행에 컬럼 "R1" 과 "R2"를 첨부합니다.

```
SELECT R1, R2, WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SALES AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
```

이것은 모든 행에 컬럼 "R1" 과 "R2"를 첨부합니다.

단계 3: 이제, 이 컬럼들을 그룹화 세트에 결합하여, 롤업 분석에서 이 컬럼들을 포함시킬 수 있습니다.

```
SELECT R1, R2,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP (WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE))),
                        (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

결과:

```
R1       R2       WEEK     DAY_WEEK  MONTH     REGION          UNITS_SOLD
-------- -------- -------- --------- --------- --------------- -----------
GROUP 1 -              13         6         - -                          27
GROUP 1 -              13         7         - -                          46
GROUP 1 -              13         -         - -                          73
GROUP 1 -              14         1         - -                          31
GROUP 1 -              14         2         - -                          43
GROUP 1 -              14         -         - -                          74
GROUP 1 -              53         1         - -                           8
GROUP 1 -              53         -         - -                           8
-        GROUP 2        -         -         3 Manitoba                   22
-        GROUP 2        -         -         3 Ontario-North               8
-        GROUP 2        -         -         3 Ontario-South              34
-        GROUP 2        -         -         3 Quebec                     40
-        GROUP 2        -         -         3 -                         104
-        GROUP 2        -         -         4 Manitoba                   17
-        GROUP 2        -         -         4 Ontario-North               1
-        GROUP 2        -         -         4 Ontario-South              14
-        GROUP 2        -         -         4 Quebec                     11
-        GROUP 2        -         -         4 -                          43
-        GROUP 2        -         -        12 Manitoba                    2
-        GROUP 2        -         -        12 Ontario-South               4
-        GROUP 2        -         -        12 Quebec                      2
-        GROUP 2        -         -        12 -                           8
-        GROUP 2        -         -         - -                         155
GROUP 1 -              -         -         - -                         155
```

단계 4: R1 및 R2는 서로 다른 그룹화 세트에서 사용되므로, R1이 결과에서 널(NULL)이 아닐 때마다 R2는 널(NULL)이고, R2가 결과에서 널(NULL)이 아닐 때마다 R1은 널(NULL)입니다. 이는 이러한 컬럼을 COALESCE 함수를 사용하여 단일 컬럼으로 병합할 수 있음을 의미합니다. 또한 두 개의 그룹 집합 결과를 유지하기 위해 ORDER BY절에서 이 컬럼을 사용할 수 있습니다.

```
SELECT COALESCE (R1,R2) AS GROUP,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
```

```
GROUP BY GROUPING SETS ((R1, ROLLUP (WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE))),
                        (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY GROUP, WEEK, DAY_WEEK, MONTH, REGION;
```

결과:

```
GROUP   WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
------- ----------- ----------- ----------- --------------- -----------
GROUP 1          13           6           - -                         27
GROUP 1          13           7           - -                         46
GROUP 1          13           -           - -                         73
GROUP 1          14           1           - -                         31
GROUP 1          14           2           - -                         43
GROUP 1          14           -           - -                         74
GROUP 1          53           1           - -                          8
GROUP 1          53           -           - -                          8
GROUP 1           -           -           - -                        155
GROUP 2           -           -           3 Manitoba                  22
GROUP 2           -           -           3 Ontario-North              8
GROUP 2           -           -           3 Ontario-South             34
GROUP 2           -           -           3 Quebec                    40
GROUP 2           -           -           3 -                        104
GROUP 2           -           -           4 Manitoba                  17
GROUP 2           -           -           4 Ontario-North              1
GROUP 2           -           -           4 Ontario-South             14
GROUP 2           -           -           4 Quebec                    11
GROUP 2           -           -           4 -                         43
GROUP 2           -           -          12 Manitoba                   2
GROUP 2           -           -          12 Ontario-South              4
GROUP 2           -           -          12 Quebec                     2
GROUP 2           -           -          12 -                          8
GROUP 2           -           -           - -                        155
```

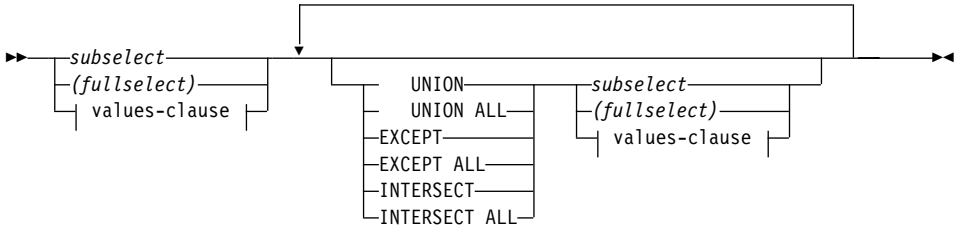
예 C8: 다음 예는 CUBE 수행시 다양한 컬럼 함수들의 이용을 보여주고 있습니다. 또한 유형변환(cast) 함수를 사용하고 있고 합당한 정밀도(precision)와 규모(scale)의 소수 결과를 제시하고 있습니다.

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD,
       MAX(SALES) AS BEST_SALE,
       CAST(ROUND(AVG(DECIMAL(SALES)),2) AS DECIMAL(5,2)) AS AVG_UNITS_SOLD
FROM SALES
GROUP BY CUBE(MONTH(SALES_DATE),REGION)
ORDER BY MONTH, REGION
```

결과:

```
MONTH       REGION          UNITS_SOLD  BEST_SALE   AVG_UNITS_SOLD
----------- --------------- ----------- ----------- --------------
          3 Manitoba                 22           7           3.14
          3 Ontario-North             8           3           2.67
          3 Ontario-South            34          14           4.25
          3 Quebec                   40          18           5.00
          3 -                       104          18           4.00
          4 Manitoba                 17           9           5.67
          4 Ontario-North             1           1           1.00
          4 Ontario-South            14           8           4.67
          4 Quebec                   11           8           5.50
          4 -                        43           9           4.78
         12 Manitoba                  2           2           2.00
         12 Ontario-South             4           3           2.00
         12 Quebec                    2           1           1.00
         12 -                         8           3           1.60
          - Manitoba                 41           9           3.73
          - Ontario-North             9           3           2.25
          - Ontario-South            52          14           4.00
          - Quebec                   53          18           4.42
          - -                       155          18           3.87
```

fullselect



values절:

값 행:

*fullselect*는 선택 명령문, INSERT문, 그리고 CREATE VIEW문의 구성요소입니다. 그것은 또한 명령문의 구성요소가 되는 특정 술어의 구성요소이기도 합니다. 술어의 구성요소인 *fullselect*를 *부속 조회*라고 합니다. 괄호로 묶인 *fullselect*를 가끔 부속 조회라고 합니다.

집합 연산자 UNION, EXCEPT, INTERSECT는 관계 연산자인 합집합, 차집합, 교집합에 해당됩니다.

*fullselect*는 결과 테이블을 지정합니다. 집합 연산자를 사용하지 않으면, *fullselect*의 결과는 지정된 부속 선택이나 values절의 결과입니다.

values절

결과 테이블에서 한 행의 각 컬럼에 대해 표현식을 사용하여 실제 값을 지정함으로써 결과 테이블을 도출합니다. 여러 행을 지정할 수 있습니다.

NULL은 복수의 *값 행*으로만 사용될 수 있고, 같은 컬럼에서 적어도 한 행은 NULL이 아니어야 합니다(SQLSTATE 42826).

*값 행*은 다음에 의해 지정됩니다.

- 단일 컬럼 결과 테이블에 대한 단일 표현식.
- 쉼표로 구분되고 괄호로 묶여 있는 n개의 표현식(또는 널(NULL)). 여기서, n은 결과 테이블의 컬럼 수입니다.

복수의 행 VALUES절에서는 각 *값 행* 표현식의 수가 같아야 합니다 (SQLSTATE 42826).

다음은 values절과 그 의미들에 대한 예입니다.

```
VALUES (1),(2),(3)          - 1컬럼의 3개의 행
VALUES 1, 2, 3              - 1컬럼의 3개의 행
VALUES (1, 2, 3)            - 3개의 컬럼에서 1행
VALUES (1,21),(2,22),(3,23) - 2개의 컬럼에서 3개의 행
```

n개의 *값 행*으로 구성된 값절, RE_1에서 RE_n은 다음과 같습니다. 여기서 n은 1보다 큽니다.

$$RE_1 \text{ UNION ALL } RE_2 \text{ . .. UNION ALL } RE_n$$

이는 각 *값 행*의 해당되는 표현식이 비교 가능해야 하고(SQLSTATE 42825), 결과의 데이터 유형이 125 페이지의 『결과 데이터 유형 규칙』에 따라 달라진다는 것을 의미합니다.

UNION 또는 **UNION ALL**

두 개의 다른 결과 테이블(R1과 R2)을 결합하여 결과 테이블을 도출합니다. UNION ALL을 지정한 경우, 결과는 R1과 R2의 모든 행으로 구성됩니다. UNION을 ALL 옵션없이 지정하면, 결과는 중복되는 행은 제거된 채로 R1이나 R2에 있는 모든 행들의 세트입니다. 그러나, 어느 경우에서든지 UNION 테이블의 각 행은 R1의 행이거나 R2의 행입니다.

EXCEPT 또는 **EXCEPT ALL**

두 개의 다른 결과 테이블(R1과 R2)을 결합하여 결과 테이블을 도출합니다. EXCEPT ALL을 지정할 경우, 결과는 R2에 해당되는 행이 없는 모든 행으

로 구성됩니다. 이 때, 중복되는 행도 포함됩니다. EXCEPT를 ALL 옵션없이 지정하면, 결과는 이 조작의 결과에서 중복되는 행은 제거된 채로 R1에만 있는 모든 행들로 구성됩니다.

INTERSECT 또는 **INTERSECT ALL**
두 개의 다른 결과 테이블(R1과 R2)을 결합하여 결과 테이블을 도출합니다. INTERSECT ALL을 지정한 경우, 결과는 R1과 R2 둘다에 있는 모든 행으로 구성됩니다. INTERSECT를 ALL 옵션없이 지정하면, 결과는 중복되는 행은 제거된 채로 R1 및 R2 둘다에 있는 행들로 구성됩니다.

결과 테이블 R1과 R2에 있는 컬럼 수는 동일해야 합니다(SQLSTATE 42826). ALL 키워드를 지정하지 않을 경우, R1 및 R2에는 255 바이트보다 큰 문자열 컬럼을 선언할 수 없고, 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, 이 유형들 중 하나에 대한 구별 유형, 또는 구조화 유형이 될 수 없습니다(SQLSTATE 42907).

결과의 컬럼들은 다음과 같이 명명됩니다.

- R의 *n*번째 컬럼과 R2의 *n*번째 컬럼에 같은 결과를 갖고 있으면, R의 *n*번째 컬럼에 결과 컬럼 이름을 갖게 됩니다.
- R1의 *n*번째 컬럼과 R2의 *n*번째 컬럼이 결과 컬럼 이름과 다를 경우, 이름이 생성됩니다. 이 이름은 ORDER BY나 UPDATE절에서 컬럼 이름으로 사용할 수 없습니다.

생성된 이름은 SQL문의 DESCRIBE를 수행하고 SQLNAME 필드를 참조하여 판별할 수 있습니다.

첫번째의 각 값이 두 번째의 해당 값과 같을 경우에만 두 행은 다른 행과 중복됩니다(중복이라고 판별되면, 두 개의 널(NULL) 값은 같은 것으로 간주됩니다.).

여러 개의 조작이 하나의 표현식에서 결합되면, 괄호 내의 조작이 먼저 수행됩니다. 괄호가 없는 경우, 모든 INTERSECT 작업이 UNION 또는 EXCEPT 조작 전에 수행되는 것을 제외하고는 왼쪽에서 오른쪽으로 수행됩니다.

다음의 예에서, R1 및 R2 테이블의 값은 왼쪽에 표시됩니다. 다른 표에서는 R1과 R2에서의 여러 가지 집합 연산의 결과 값을 보여줍니다.

R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTER-SECT ALL	INTER-SECT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					
4		3					
5		3					
		3					
		3					
		3					
		4					
		4					
		4					
		5					

결과 컬럼의 데이터 유형이 판별되는 방법에 대한 규칙은 125 페이지의 『결과 데이터 유형 규칙』에서 참조하십시오.

문자열(string) 컬럼의 변환이 처리되는 방법에 관한 규칙에 대해서는, 129 페이지의 『문자열 변환 규칙』에서 참조하십시오.

fullselect의 예

예 1: EMPLOYEE 테이블에서 모든 컬럼과 행을 선택합니다.

```
SELECT * FROM EMPLOYEE
```

예 2: 부서 번호가 'E'로 시작하는 EMPLOYEE 테이블에 있거나, 또는 프로젝트 번호(PROJNO)가 'MA2100', 'MA2110' 또는 'MA2112'인 EMP_ACT 테이블에서 프로젝트에 지정된 모든 사원의 사원 번호(EMPNO)를 나열합니다.

```
  SELECT EMPNO
   FROM EMPLOYEE
    WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

예 3: 예 2에서와 같은 조회를 수행하는데, 추가로 'emp'가 있는 EMPLOYEE 테이블의 행과 'emp_act'가 있는 EMP_ACT 테이블의 행을 "태그" 처리합니다. 예 2에서의 결과와는 달리, 이 조회에서는 동일한 EMPNO를 두 번 이상 리턴할 것이며, 관련된 "태그"를 이용하여 행이 속해 있는 테이블을 식별합니다.

```
SELECT EMPNO, 'emp'
   FROM EMPLOYEE
   WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO, 'emp_act' FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

예 4: 예 2와 같은 조회를 만들고, 중복 행이 제거되지 않도록 UNION ALL만 사용합니다.

```
SELECT EMPNO
    FROM EMPLOYEE
    WHERE WORKDEPT LIKE 'E%'
UNION ALL
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

예 5: 예 3과 같은 조회를 만들고, 현재 테이블에 없는 두 명의 사원을 추가로 포함시키고 이들 행에 ″new″ 태그를 붙입니다.

```
SELECT EMPNO, 'emp'
   FROM EMPLOYEE
   WHEREWORKDEPTLIKE 'E%'
UNION
SELECT EMPNO, 'emp_act'
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
UNION
   VALUES ('NEWAAA', 'new'), ('NEWBBB', 'new')
```

예 6: 이 EXCEPT의 예는 T1에는 있으나 T2에 없는 모든 행을 생성합니다.

```
(SELECT * FROM T1)
EXCEPT ALL
(SELECT * FROM T2)
```

어떤 널(NULL) 값도 포함되어 있지 않으면, 이 예는 다음과 같은 결과를 리턴합니다.

```
SELECT ALL *
   FROM T1
   WHERE NOT EXISTS (SELECT * FROM T2
                     WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

예 7: INTERSECT의 이 예는 T1과 T2 테이블에 있는 행을 모두 생성하지만 중복된 행은 제거합니다.

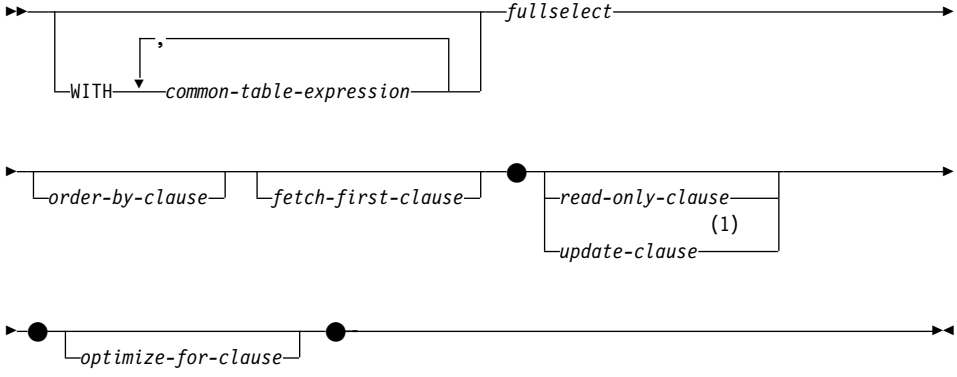
```
(SELECT * FROM T1)
INTERSECT
(SELECT * FROM T2)
```

어떤 널(NULL) 값도 포함되어 있지 않으면, 이 예는 다음과 같은 결과를 리턴합니다.

```
SELECT DISTINCT * FROM T1
   WHERE EXISTS (SELECT * FROM T2
                 WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

여기서, C1, C2 등은 T1 및 T2의 컬럼을 나타냅니다.

select문



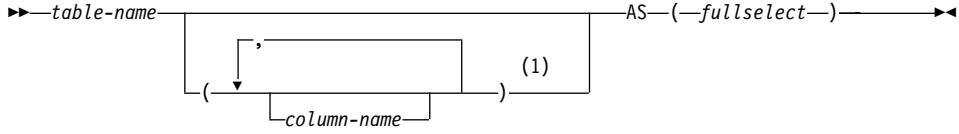
주:

1 update절과 order-by절 모두가 동일한 select문에 지정될 수 없습니다.

*select-statement*는 DECLARE CURSOR문에서 직접 지정되거나, DECLARE CURSOR문에서 준비된 후 참조될 수 있는 조회 양식입니다. 명령행 처리기(CLP)(또는 유사한 도구)를 사용해서 동적 SQL문을 실행할 수도 있으며 결과 테이블이 사용자의 화면에 표시됩니다. 어느 경우든, *select-statement*에 의해 지정된 테이블은 fullselect의 결과입니다.

별명(nickname)을 참조하는 *select-statement*는 DECLARE CURSOR문에서 직접 지정될 수 없습니다.

공통 테이블 표현식



주:

1 공통 테이블 표현식이 순환되는(recursive) 경우나 fullselect 결과로 중복되는 컬럼 이름이 발생할 경우, 컬럼 이름을 지정해야 합니다.

*공통 테이블 표현식*을 사용하여, 뒤에 오는 fullselect의 FROM절에 하나의 테이블 이름으로 지정할 수 있는 *테이블 이름*으로 결과 테이블을 정의할 수 있습니다. 복수의 공통 테이블 표현식은 단일 WITH 키워드 다음에 지정할 수 있습니다. 지정된 각 공통 테이블 표현식은 후속 공통 테이블 표현식의 FROM절에서 이름으로 참조될 수도 있습니다.

컬럼 목록이 지정되면, fullselect의 결과 테이블에 있는 컬럼 수만큼의 이름으로 구성됩니다. 각 *컬럼 이름*은 고유해야 하고 규정화되어서는 안 됩니다. 이 컬럼 이름들을 지정하지 않으면, 이름들은 공통 테이블 표현식을 정의하는 데 사용되는 fullselect의 선택 목록에서 도출됩니다.

*공통 테이블 표현식*의 *테이블 이름*은 같은 명령문 내에 있는 다른 공통 테이블 표현식 *테이블 이름*과 반드시 달라야 합니다(SQLSTATE 42726). INSERT문에 공통 테이블 표현식이 지정되면, *테이블 이름*은 삽입 오브젝트인 테이블이나 뷰(view)의 이름과 같을 수 없습니다(SQLSTATE 42726). 공통 테이블 표현식 *테이블 이름*은 fullselect를 통해 임의의 FROM절에서 테이블 이름으로 지정할 수 있습니다. 공통 테이블 표현식의 *테이블 이름*은 동일한 규정화 이름으로 임의의 기존 테이블, 뷰(view) 또는 별명(카탈로그에 있는)을 대체할 수 있습니다.

같은 명령문에 둘 이상의 공통 테이블 표현식이 정의되어 있으면, 공통 테이블 표현식 사이의 순환 참조가 허용되지 않습니다(SQLSTATE 42835). *순환 참조(cyclic reference)*는 두 개의 공통 테이블 표현식인 *dt1* 및 *dt2*가 작성될 경우, *dt1*은 *dt2*를 참조하고 *dt2*는 *dt1*을 참조합니다.

또한, *공통 테이블 표현식*은 CREATE VIEW 및 INSERT문에서 선택적으로 fullselect 앞에 옵니다.

다음의 경우 *공통 테이블 표현식*을 사용할 수 있습니다.

- 뷰(view) 작성을 피하기 위해 뷰 대신에(범용 뷰가 필요하지 않으며 갱신사항이 위치지정되지 않거나 삭제 내용이 사용되지 않을 때) 사용하려는 경우
- 스칼라 부속 선택이나, 결정적이지 않거나 외부 조치를 갖는 함수에서 파생된 컬럼별로 그룹화하려는 경우
- 원하는 결과 테이블이 호스트 변수를 기초로 할 경우
- 같은 결과 테이블이 *fullselect*에서 공유되어야 할 경우
- 결과가 순환을 사용하여 도출되어야 할 경우

공통 테이블 표현식 순환(recursion)의 *fullselect*에 FROM절에서의 그 자체에 대한 참조가 있으면, 공통 테이블 표현식은 *순환 공통 테이블 표현식(recursive common table expression)*입니다. 순환을 사용하는 조회는 부품표(BOM), 예약 시스템 및 네트워크 계획과 같은 응용프로그램을 지원할 때 유용합니다. 예를 들어, 1493 페이지의 『부록M. 순환 예: 부품표(BOM: Bill of Materials)』에서 참조하십시오.

다음은 참인 순환 공통 테이블 표현식이어야 합니다.

- 순환 주기의 부분인 fullselect는 SELECT나 SELECT ALL로 시작해야 합니다. SELECT DISTINCT의 사용은 허용되지 않습니다(SQLSTATE 42925). 또한, union은 UNION ALL을 사용해야 합니다(SQLSTATE 42925).
- 컬럼 이름은 공통 테이블 표현식의 *테이블 이름* 다음에 지정해야 합니다(SQLSTATE 42908).
- 첫번째 union의 첫번째 fullselect(초기설정 fullselect)에 FROM절의 공통 테이블 표현식 컬럼에 대한 참조가 포함되어서는 안 됩니다(SQLSTATE 42836).
- 공통 테이블 표현식의 컬럼 이름이 대화식 fullselect에서 참조된 경우, 초기설정 fullselect에 근거하여 컬럼에 대한 데이터 유형, 길이 및 코드 페이지가 결정됩니다. 반복 fullselect의 해당 컬럼은 초기설정 fullselect에 따라 결정된 데이터 유형 및 길이와 동일해야 하며, 코드 페이지는 일치해야 합니다(SQLSTATE 42825). 그러나, 문자열 유형인 경우, 두 데이터 유형의 길이는 서로 다를 수

있습니다. 이 경우, 반복 fullselect 내에 있는 컬럼은 항상 초기설정 fullselect에서 결정된 길이에 지정할 수 있는 길이를 가져야 합니다.

- 순환 주기의 일부인 각 fullselect에는 모든 컬럼 함수, GROUP-BY절 또는 HAVING절이 포함되어서는 안 됩니다(SQLSTATE 42836).

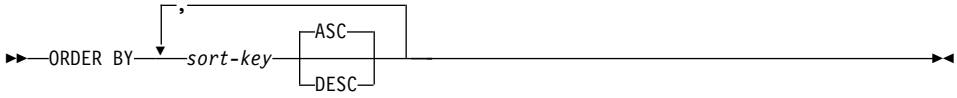
  이 fullselect들의 FROM절에는 순환 주기의 부분이 되는 공통 테이블 표현식에 대한 최소한 하나의 참조를 포함해야 합니다(SQLSTATE 42836).
- 부속 조회(스칼라 또는 규정화된)는 순환 주기의 부분이 아니어야 합니다(SQLSTATE 42836).

순환 공통 테이블 표현식을 개발할 때, 무한 순환 주기(루프)가 만들어질 수도 있습니다. 순환 주기가 종료되는지 점검하십시오. 이것은 연관된 데이터가 순환적일 때 특히 중요합니다. 순환 공통 테이블 표현식에는 무한 루프를 방지하는 술어가 포함되어야 합니다. 순환 공통 테이블 표현식에는 다음이 포함될 것으로 예상됩니다.

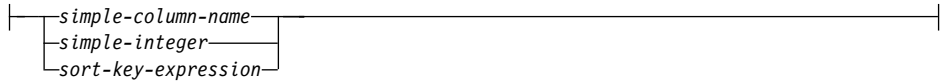
- 반복 fullselect에서, 상수로 증가되는 정수 컬럼
- ″counter_col < constant″ 또는 ″counter _col < :hostvar″ 양식에서 반복 fullselect의 Where 절에 있는 술어

순환 공통 테이블 표현식에서 이 구문을 찾을 수 없으면 경고가 발행됩니다(SQLSTATE 01605).

order-by절



sort-key:



ORDER BY절은 결과 테이블에서 행의 순서를 지정합니다. 단일 정렬 스펙(연관 방향을 가진 하나의 *sort-key*)이 식별되면, 그 정렬 스펙의 값에 따라 행이 정렬됩니다. 하나 이상의 정렬 스펙이 확인되는 경우, 첫번째 정렬 스펙의 값으로 행의 순서가 정해진 다음 두 번째 정렬 스펙, 세번째 정렬 스펙 등의 순으로 행의 순서가 정해집니다. 각 *정렬 키*의 길이 속성은 문자 컬럼의 경우 255자를, 그래픽 컬럼의 경우 127자보다 크면 안되며(SQLSTATE 42907), 데이터 유형은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, 이 유형들 중 하나에 대한 구별 유형, 또는 구조화 유형이 될 수 없습니다 (SQLSTATE 42907).

선택 목록에 있는 명명된 컬럼은 *simple-integer* 또는 *simple-column-name*인 *sort-key*로 식별될 수도 있습니다. 선택 목록의 명명되지 않은 컬럼은 선택 목록의 표현식과 일치하는 *단순 정수* 또는 어떤 경우 *정렬 키 표현식*으로 식별되어야 합니다(*정렬 키 표현식*의 세부사항 참조). AS절을 지정하지 않고, 상수, 연산자가 있는 표현식 또는 함수로부터 도출된 경우, 컬럼은 명명되지 않습니다.[54]

순서화는 『제3장 언어 요소』에 설명된 비교 규칙에 따라 수행됩니다. 널(NULL) 값은 모든 다른 값보다 높습니다. ORDER BY절이 완전히 행들을 순서화하지 않으면, 모든 식별된 컬럼의 중복 값을 갖는 행들은 임의의 순서로 표시됩니다.

54. 집합 연산자(UNION, INTERSECT 또는 EXCEPT)를 포함하는 fullselect에 대해 결과 컬럼의 이름을 판별하기 위한 규칙은 476 페이지의 『fullselect』에 나와 있습니다.

단순 컬럼 이름

보통 결과 테이블의 컬럼을 식별합니다. 이 경우, *단순 컬럼 이름*은 선택 목록에 있는 명명된 컬럼 이름이어야 합니다.

*단순 컬럼 이름*은 조회가 부속 선택인 경우, FROM절에서 식별된 테이블, 뷰 또는 중첩 테이블의 컬럼 이름도 식별할 수 있습니다. 부속 선택이 다음과 같을 경우 오류가 발생합니다.

- 선택절에서 DISTINCT를 지정합니다(SQLSTATE 42822).
- 그룹화된 결과를 생성하며 *단순 컬럼 이름*이 *그룹 표현식*이 아닙니다(SQLSTATE 42803).

결과의 순서를 정하기 위해 사용되는 컬럼을 결정하는 데 대해서는 ″정렬 키의 컬럼 이름″에 설명됩니다(488 페이지의 『주』 참조).

단순 정수

0보다 커야 하고, 결과 테이블의 컬럼 수보다 작아야 합니다(SQLSTATE 42805). 정수 *n*은 결과 테이블의 *n*번째 컬럼을 식별합니다.

정렬 키 표현식

단순한 컬럼 이름이나 미지정 정수 상수가 아닌 표현식입니다. 이러한 형태의 정렬 키를 사용하려면 순서가 적용되는 조회는 *부속 선택*이어야 합니다. *정렬 키 표현식*에는 상관 스칼라 fullselect(SQLSTATE 42703) 또는 외부 조치를 갖는 함수(SQLSTATE 42845)가 포함될 수 없습니다.

정렬 키 표현식 내의 모든 컬럼 이름은 ″정렬 키의 컬럼 이름″에 기술된 규칙을 따라야 합니다(488 페이지의 『주』 참조).

지정할 수 있는 표현식에 대한 제한이 추가되는 특별한 경우도 많습니다.

- DISTINCT는 부속 선택의 SELECT절에 지정됩니다(SQLSTATE 42822). 정렬 키 표현식은 부속 선택의 선택 목록 표현식과 정확히 일치해야 합니다(스칼라 fullselect와는 절대 일치하지 않음).
- 부속 선택은 그룹화됩니다(SQLSTATE 42803).

  정렬 키 표현식은

  – 부속 선택의 선택 목록 표현식일 수 있습니다.

  – 부속 선택의 GROUP BY절에 있는 *그룹 표현식*을 포함할 수 있습니다.

– 컬럼 함수, 상수 또는 호스트 변수를 포함할 수 있습니다.

ASC

컬럼의 값을 오름차순으로 사용합니다. 이것은 기본값입니다.

DESC

컬럼의 값을 내림차순으로 사용합니다.

주

- **정렬 키의 컬럼 이름**:
  – 컬럼 이름이 규정됩니다.

    조회는 *부속 선택*이어야 합니다(SQLSTATE 42877). 컬럼 이름은 부속 선택의 FROM절에 있는 테이블, 뷰 또는 중첩 테이블의 컬럼을 명확하게 식별해야 합니다(SQLSTATE 42702). 컬럼의 값은 정렬 스펙의 값을 계산하는 데 사용됩니다.

  – 컬럼 이름이 규정되지 않습니다.

    - 조회는 부속 선택입니다.

      컬럼 이름이 결과 테이블에 있는 하나 이상의 컬럼 이름과 일치하는 경우, 그 컬럼 이름은 순서화 부속 선택의 FROM절에 있는 테이블, 뷰 또는 중첩 테이블의 컬럼을 명확하게 식별해야 합니다(SQLSTATE 42702). 컬럼 이름이 한 컬럼과 일치하는 경우, 그 컬럼은 정렬 스펙의 값을 계산하는 데 사용됩니다. 컬럼 이름이 결과 테이블의 컬럼과 일치하지 않는 경우, 이는 선택문의 fullselect에 있는 FROM절의 테이블, 뷰 또는 중첩 테이블의 컬럼을 명확하게 식별해야 합니다(SQLSTATE 42702).

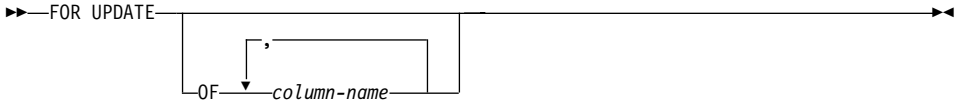
    - 조회는 부속 선택이 아닙니다(합병, 예외 또는 교차와 같은 조작을 포함합니다.).

      컬럼 이름이 결과 테이블의 하나 이상의 컬럼 이름과 일치해서는 안 됩니다(SQLSTATE 42702). 컬럼 이름이 결과 테이블의 한 컬럼과 정확히 일치해야 하며(SQLSTATE 42707) 이 컬럼이 정렬 스펙의 값을 계산하는 데 사용됩니다.

    규정된 컬럼 이름에 대해 151 페이지의 『모호성을 회피하기 위한 컬럼 이름 규정자』에서 자세한 정보를 참조하십시오.

- **제한사항**: *정렬 키 표현식* 또는 컬럼이 선택 목록에 있지 않은 *단순 컬럼 이름*을 사용하면 정렬시 사용되는 임시 테이블에 컬럼이나 표현식이 추가됩니다. 이는 또 테이블에 들어갈 컬럼의 수나 테이블에서의 행의 크기를 제한하게 됩니다. 이 한도를 초과하면, 임시 테이블을 통해 정렬 연산을 수행해야 하는 경우 오류가 발생합니다.

update절

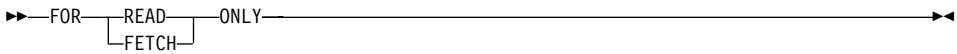


FOR UPDATE절은 그 다음의 Positioned UPDATE문에서 갱신될 수 있는 컬럼을 식별합니다. 각 *컬럼 이름*은 규정하지 않아야 하고, fullselect의 첫번째 FROM에서 식별된 테이블이나 뷰의 컬럼을 식별해야 합니다. FOR UPDATE절이 컬럼 이름 없이 지정된 경우, fullselect의 첫번째 FROM절에서 식별되는 테이블 또는 뷰의 갱신 가능한 모든 컬럼이 포함됩니다.

다음 중 하나가 만족될 경우 FOR UPDATE 절은 사용할 수 없습니다.

- 선택문과 연관된 커서는 삭제할 수 없습니다(955 페이지의 『주』 참조).
- 선택된 컬럼 중 하나는 카탈로그 테이블의 불가능 컬럼이며, FOR UPDATE절은 이 컬럼을 제거하는 데 사용되지 않습니다.

read-only절



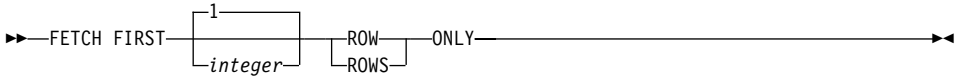
FOR READ ONLY절은 결과 테이블이 읽기 전용이어서 커서가 Positioned UPDATE 및 DELETE문에서 참조될 수 없음을 나타냅니다. FOR FETCH ONLY는 같은 의미를 갖고 있습니다.

일부 결과 테이블은 원래 읽기 전용입니다. (예를 들어, 테이블은 읽기 전용 뷰(view)를 기초로 합니다.) 그러한 테이블에 대해 FOR READ ONLY를 계속 지정할 수 있으나, 그 지정은 효과가 없습니다.

갱신과 삭제가 허용되는 결과 테이블의 경우, FOR READ ONLY(또는 FOR FETCH ONLY)를 지정하면, 데이터베이스 관리 프로그램이 블록킹을 수행할 수 있도록 하여 독점 잠금을 피할 수 있도록 함으로써 FETCH 작업의 성능을 향상시킬 수 있습니다. 예를 들어, FOR READ ONLY 또는 ORDER BY절이 없는 동적 SQL문을 포함하는 프로그램에서, 데이터베이스 관리 프로그램은 FOR UPDATE절이 지정되었을 때처럼 열릴 수 있습니다. 그러므로, FOR READ ONLY절을 사용하면 조회가 Positioned UPDATE 또는 DELETE문에서 사용되는 경우를 제외하고 성능을 향상시킵니다.

읽기 전용 결과 테이블은 원래 읽기 전용인지, 아니면 FOR READ ONLY(FOR FETCH ONLY)로 지정되었는지에 관계없이 Positioned UPDATE 또는 DELETE문에서 참조되면 안 됩니다. 읽기 전용의 갱신 가능한 커서에 대해 952 페이지의 『DECLARE CURSOR』에서 좀더 자세한 정보를 참조하십시오.

fetch-first절

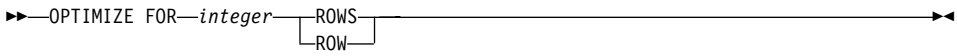


*fetch-first절*은 검색할 수 있는 행의 최대 수를 설정합니다. 이렇게 하면, 이 절이 지정되지 않을 때 결과 테이블에 있을 수 있는 행의 수에 관계없이, 응용프로그램이 *정수* 행 이상은 검색하지 않으려 한다는 것을 데이터베이스 관리 프로그램이 알게 됩니다. *정수* 행을 벗어나서 페치하려는 시도는 데이터의 정상 종료와 같은 방법으로 처리됩니다(SQLSTATE 02000). *정수* 값은(제로가 아닌) 양수여야 합니다.

결과 테이블을 첫번째 *정수* 행으로 제한하면 성능이 향상될 수 있습니다. 데이터베이스 관리 프로그램은 첫번째 *정수* 행을 결정하면 조회 처리를 중지합니다. *페치 우선 절*과 *optimize-for절* 모두가 지정되는 경우, 이 절에서 낮은 *정수* 값을 사용하여 통신 버퍼 크기에 영향을 주게 됩니다. 이 값들은 최적화 목적과는 독립적으로 고려됩니다.

Select문에 fetch-first절을 지정하면 커서가 사용불가능하게 (읽기 전용으로) 됩니다. 이 절은 FOR UPDATE절과 함께 지정할 수 없습니다.

optimize-for절



OPTIMIZE FOR절은 *select*문의 특수 처리를 요구합니다. 절이 생략되는 경우, 결과 테이블의 모든 행이 검색되는 것으로 간주됩니다. 절이 지정되는 경우, 검색되는 행의 수는 *n*을 초과하지 않으며, 여기서 *n*은 *정수*에 대한 값입니다. *n*의 값은 양수여야 합니다. OPTIMIZE FOR절은 *n* 행이 검색된다는 가정에 따라 조회 최적화에 영향을 미칩니다. 또한, 블록화되는 커서의 경우, 이 절은 각 블록에서 리턴될 행 수에 영향을 줍니다. 즉, 최대 *n*개의 행이 각 블록에서 리턴됩니다. *페치 우선 절*과 *optimize-for절* 모두가 지정되는 경우, 이 절에서 낮은 정수 값을 사용하여 통신 버퍼 *크기*에 영향을 주게 됩니다. 이 값들은 최적화 목적과는 독립적으로 고려됩니다.

이 절은 페치될 수 있는 행 수를 제한하거나 성능 외의 다른 방법에서의 결과에 영향을 미치지 않습니다. OPTIMIZE FOR *n* ROWS는 *n*개의 행 수보다 적게 검색되면 성능이 향상될 수 있으나, *n*개 이상의 행 수가 검색되면 성능은 떨어집니다.

*n*의 값에 행 *크기*를 곱한 값이 통신 버퍼의 *크기*를 초과하면, [55] OPTIMIZE FOR 절은 데이터 버퍼에 영향을 주지 않습니다.

55. 통신 버퍼의 *크기*는 RQRIOBLK나 ASLHEAPSZ 구성 매개변수에 의해 정의됩니다. *관리 안내서*에서 자세한 내용을 참조하십시오.

select문의 예

예 1: EMPLOYEE 테이블에서 모든 컬럼과 행을 선택합니다.

```
SELECT * FROM EMPLOYEE
```

예 2: PROJECT 테이블에서 프로젝트 이름(PROJNAME), 시작일(PRSTDATE) 및 종료일(PRENDATE)을 선택합니다. 가장 최근의 날짜가 처음으로 나타나는, 종료일별로 결과 테이블을 순서화합니다.

```
SELECT PROJNAME, PRSTDATE, PRENDATE
               FROM PROJECT
 ORDER BY PRENDATE DESC
```

예 3: EMPLOYEE 테이블에 있는 모든 부서에 대해, 부서 번호(WORKDEPT)와 부서의 평균 급여(SALARY)를 선택합니다. 결과 테이블을 평균 부서 급료의 오름차순으로 배열합니다.

```
SELECT WORKDEPT, AVG(SALARY)
   FROM EMPLOYEE
   GROUP BY WORKDEPT
    ORDER BY 2
```

예 4: PROJECT 테이블에서 시작 날짜(PRSTDATE)와 종료 날짜(PRENDATE) 컬럼을 갱신하기 위해 C 프로그램에서 사용될 UP_CUR 커서를 선언합니다. 프로그램은 각 행에 대한 프로젝트 번호(PROJNO) 값과 함께 이 두 값을 모두 승인해야 합니다.

```
EXEC SQL  DECLARE UP_CUR CURSOR FOR
            SELECT PROJNO, PRSTDATE, PRENDATE
              FROM PROJECT
              FOR UPDATE OF PRSTDATE, PRENDATE;
```

예 5: 이 예에서는 표현식 SAL+BONUS+COMM을 TOTAL_PAY로 명명합니다.

```
SELECT SALARY+BONUS+COMM AS TOTAL_PAY
  FROM EMPLOYEE
  ORDER BY TOTAL_PAY
```

예 6: 영업 담당자의 사원 번호와 급여 및 이들 부서의 평균 급여와 인원수를 판별합니다. 또한, 가장 높은 평균 급여와 함께 부서의 평균 급여를 나열합니다.

이 경우에 공통 테이블 표현식을 사용하면, 정규 뷰(view)로 DINFO 뷰를 작성할 때의 오버헤드가 줄어듭니다. 명령문 준비시, 영업 대표들의 부서에 해당되는 행들인 fullselect의 나머지 문맥이 뷰에서 고려되어야 하므로, 뷰에 대한 카탈로그 액세스를 피할 수 있습니다.

```
WITH
   DINFO (DEPTNO, AVGSALARY, EMPCOUNT) AS
       (SELECT OTHERS.WORKDEPT, AVG(OTHERS.SALARY), COUNT(*)
         FROM EMPLOYEE OTHERS
         GROUP BY OTHERS.WORKDEPT
       ),
   DINFOMAX AS
       (SELECT MAX(AVGSALARY) AS AVGMAX FROM DINFO)
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY,
       DINFO.AVGSALARY, DINFO.EMPCOUNT, DINFOMAX.AVGMAX
 FROM EMPLOYEE THIS_EMP, DINFO, DINFOMAX
 WHERE THIS_EMP.JOB = 'SALESREP'
 AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

제6장 SQL문

이 장에는 구문 도표, 어휘 설명, 규칙, 그리고 SQL문 사용에 대한 예를 수록하고 있습니다.

표 18. SQL문

SQL문	함수	페이지
ALTER BUFFERPOOL	버퍼 풀의 정의를 변경합니다.	508
ALTER NICKNAME	별명의 정의를 변경합니다.	511
ALTER NODEGROUP	노드 그룹의 정의를 변경합니다.	515
ALTER SERVER	서버의 정의를 변경합니다.	519
ALTER TABLE	테이블의 정의를 변경합니다.	524
ALTER TABLESPACE	테이블 공간의 정의를 변경합니다.	554
ALTER TYPE(구조화)	구조화 유형의 정의를 변경합니다.	561
ALTER USER MAPPING	사용자 권한 부여 맵핑의 정의를 변경합니다.	569
ALTER VIEW	참조 유형 컬럼을 변경하여 영역에 추가함으로써 뷰의 정의를 변경합니다.	572
BEGIN DECLARE SECTION	호스트 변수 선언 세션의 시작부분을 표시합니다.	574
CALL	저장 프로시듀어를 호출합니다.	577
CLOSE	커서를 닫습니다.	586
COMMENT ON	오브젝트의 설명을 주석으로 대체하거나 주석을 추가합니다.	588
COMMIT	작업 단위(UOW)를 종료하고 그 작업 단위(UOW)에서 시행한 데이터베이스 변경을 확약합니다.	601
복합 SQL(포함)	하나 이상의 SQL문을 실행 가능한 블록에 결합합니다.	603
CONNECT(유형 1)	원격 작업 단위의 규칙에 따라 응용프로그램 서버(AS)에 연결합니다.	608
CONNECT(유형 2)	응용프로그램 지정 분산 작업 단위(DUOW)의 규칙에 따라 응용프로그램 서버(AS)에 연결합니다.	618
CREATE ALIAS	테이블, 뷰 또는 다른 별명에 대한 별명을 정의합니다.	627
CREATE BUFFERPOOL	새로운 버퍼 풀을 생성합니다.	631
CREATE DISTINCT TYPE	개별 데이터 유형을 정의합니다.	635
CREATE EVENT MONITOR	모니터할 데이터베이스에서 이벤트를 지정합니다.	642
CREATE FUNCTION	사용자 정의 함수를 등록합니다.	653

표 18. SQL문 (계속)

SQL문	함수	페이지
CREATE FUNCTION(외부 스칼라)	사용자 정의 외부 스칼라 함수를 등록합니다.	655
CREATE FUNCTION(외부 테이블)	사용자 정의 외부 테이블 함수를 등록합니다.	685
CREATE FUNCTION(OLE DB 외부 테이블)	사용자 정의 OLE DB 외부 테이블 함수를 등록합니다.	704
CREATE FUNCTION (소스 또는 템플리트)	사용자 정의 전래 함수를 등록합니다.	714
CREATE FUNCTION (SQL 스칼라, 테이블 또는 행)	사용자 정의 SQL 함수를 등록 및 정의합니다.	726
CREATE FUNCTION MAPPING	함수 맵핑을 정의합니다.	735
CREATE INDEX	테이블에서 색인을 정의합니다.	740
CREATE INDEX EXTENSION	구조화 또는 구별 유형의 컬럼이 있는 테이블에서 색인으로 사용할 확장 오브젝트를 정의합니다.	749
CREATE METHOD	메소드 내용을 이전에 정의된 메소드 스펙과 연관시킵니다.	758
CREATE NICKNAME	별명을 정의합니다.	764
CREATE NODEGROUP	노드 그룹을 정의합니다.	768
CREATE PROCEDURE	저장 프로시듀어를 등록합니다.	771
CREATE SCHEMA	스키마를 정의합니다.	791
CREATE SERVER	데이터 소스를 연합 데이터베이스에 정의합니다.	795
CREATE TABLE	테이블을 정의합니다.	800
CREATE TABLESPACE	테이블 공간을 정의합니다.	861
CREATE TRANSFORM	변환 함수를 정의합니다.	872
CREATE TRIGGER	트리거를 정의합니다.	879
CREATE TYPE(구조화)	구조화 데이터 유형을 정의합니다.	893
CREATE TYPE MAPPING	데이터 유형간의 맵핑을 정의합니다.	922
CREATE USER MAPPING	사용자 권한 부여간의 맵핑을 정의합니다.	928
CREATE VIEW	하나 이상의 테이블, 뷰 또는 별명의 뷰를 정의합니다.	931
CREATE WRAPPER	랩퍼를 등록합니다.	949
DECLARE CURSOR	SQL 커서를 정의합니다.	952
DECLARE GLOBAL TEMPORARY TABLE	전역 임시 테이블을 정의합니다.	958
DELETE	테이블에서 두 개 이상의 행을 삭제합니다.	968

표 18. SQL문 (계속)

SQL문	함수	페이지
DESCRIBE	준비된 SELECT문의 결과 컬럼을 설명합니다.	974
DISCONNECT	활동중인 작업 단위가 없을 때, 하나 이상의 연결을 종료합니다.	980
DROP	데이터베이스에서 오브젝트를 삭제합니다.	984
END DECLARE SECTION	호스트 변수 선언 구획의 끝을 표시합니다.	1014
EXECUTE	준비된 SQL문을 실행합니다.	1016
EXECUTE IMMEDIATE	SQL문을 준비하고 실행합니다.	1022
EXPLAIN	선택된 액세스 플랜에 관한 정보를 보관합니다.	1025
FETCH	호스트 변수에 행의 값을 지정합니다.	1031
FLUSH EVENT MONITOR	이벤트 모니터의 사용중인 내부 버퍼를 씁니다.	1035
FREE LOCATOR	위치 지정자 변수와 해당 값 사이의 연관을 제거합니다.	1037
GRANT(데이터베이스 권한)	전체 데이터베이스에 대해 권한을 권한 부여합니다.	1039
GRANT(색인 특권)	데이터베이스에 있는 색인에 대해 CONTROL 특권을 권한 부여합니다.	1043
GRANT(패키지 특권)	데이터베이스에 있는 패키지에 대해 특권을 권한 부여합니다.	1045
GRANT(스키마 특권)	스키마에 대해 특권을 권한 부여합니다.	1048
GRANT(서버 특권)	특정 데이터 소스를 조회할 특권을 권한 부여합니다.	1052
GRANT(테이블, 뷰 또는 별명 특권)	테이블, 뷰 및 별명(nickname)에 대한 특권을 권한 부여합니다.	1054
GRANT(테이블 공간 권한 취소)	테이블 공간에 대해 특권을 권한 부여합니다.	1064
INCLUDE	원시 프로그램에 코드나 선언을 삽입합니다.	1067
INSERT	테이블에 두 개 이상의 행을 삽입합니다.	1069
LOCK TABLE	테이블 변경시의 동시 프로세스를 방지하거나 테이블 사용시의 동시 프로세스를 방지합니다.	1079
OPEN	FETCH문이 발행될 때 값을 검색하기 위해 사용되는 커서를 준비합니다.	1081
PREPARE	실행을 위해 SQL문(선택적 매개변수와 함께)을 준비합니다.	1087
REFRESH TABLE	요약 테이블의 데이터를 새로 고칩니다.	1097
RELEASE(연결)	하나 이상의 연결을 해제 보류 상태로 둡니다.	1098
RELEASE SAVEPOINT	트랜잭션 내에서 저장 지점을 해제합니다.	1100
RENAME TABLE	기존 테이블을 재명명합니다.	1101
RENAME TABLESPACE	기본 테이블 공간의 이름을 바꿉니다.	1103
REVOKE(데이터베이스 권한)	전체 데이터베이스에서 권한을 권한 취소합니다.	1105
REVOKE(색인 특권)	제공된 색인에 대해 CONTROL 특권을 권한 취소합니다.	1109
REVOKE(패키지 특권)	데이터베이스의 주어진 패키지에서 특권을 권한 취소합니다.	1112

표 18. SQL문 (계속)

SQL문	함수	페이지
REVOKE(스키마 특권)	스키마에 대해 특권을 권한 취소합니다.	1115
REVOKE(서버 특권)	특정 데이터 소스를 조회할 특권을 권한 취소합니다.	1118
REVOKE(테이블, 뷰 또는 별명 특권)	제공된 테이블, 뷰 또는 별명으로부터 특권을 권한 취소합니다.	1120
REVOKE(테이블 공간 권한 취소)	주어진 테이블 공간에서 USE 특권을 취소합니다.	1127
ROLLBACK	작업 단위(UOW)를 종료하고 작업 단위(UOW)에서 시행한 데이터베이스 변경을 취소합니다.	1130
SAVEPOINT	트랜잭션 내에서 저장 지점을 설정합니다.	1133
SELECT INTO	단 한 행의 결과 테이블을 지정하고 호스트 변수에 값을 지정합니다.	1136
SET CONNECTION	연결의 상태를 휴면(dormant)에서 현재로 변경하여, 지정된 위치를 서버로 만듭니다.	1138
SET CURRENT DEFAULT TRANSFORM GROUP	CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터의 값을 변경합니다.	1141
SET CURRENT DEGREE	CURRENT DEGREE 특수 레지스터의 값을 변경합니다.	1143
SET CURRENT EXPLAIN MODE	CURRENT EXPLAIN MODE 특수 레지스터의 값을 변경합니다.	1145
SET CURRENT EXPLAIN SNAPSHOT	CURRENT EXPLAIN SNAPSHOT 특수 레지스터의 값을 변경합니다.	1148
SET CURRENT PACKAGESET	선택한 패키지에 대한 스키마 이름을 설정합니다.	1150
SET CURRENT QUERY OPTIMIZATION	CURRENT QUERY OPTIMIZATION 특수 레지스터의 값을 변경합니다.	1152
SET CURRENT REFRESH AGE	CURRENT REFRESH AGE 특수 레지스터의 값을 변경합니다.	1156
SET EVENT MONITOR STATE	이벤트 모니터를 활성화하거나 비활성화합니다.	1158
SET INTEGRITY	점검 보류 상태를 설정하고 제한사항 위반에 대해 데이터를 점검합니다.	1160
SET PASSTHRU	데이터 소스의 원시 SQL을 데이터 소스에 직접 제출하기 위한 세션을 엽니다.	1172
SET PATH	CURRENT PATH 특수 레지스터의 값을 변경합니다.	1174
SET SCHEMA	CURRENT SCHEMA 특수 레지스터의 값을 변경합니다.	1177
SET SERVER OPTION	서버 옵션 설정값을 설정합니다.	1180
SET 전이 변수	NEW 임시 변수에 값을 지정합니다.	1182
SIGNAL SQLSTATE	오류 신호를 보냅니다.	1187

표 18. SQL문 (계속)

SQL문	함수	페이지
UPDATE	테이블의 하나 이상의 행에서 하나 이상의 컬럼에 대한 값을 갱신합니다.	1189
VALUES INTO	단 한 행의 결과 테이블을 지정하고 호스트 변수에 값을 지정합니다.	1202
WHENEVER	SQL 리턴 코드를 기준으로 취해질 조치를 정의합니다.	1204

SQL문이 호출되는 방법

이 장에 설명된 SQL문은 *실행 가능* 또는 *실행 불가능*으로 분류됩니다. 각 명령문의 설명에서 호출 절은 명령문이 실행 가능한지를 나타냅니다.

*실행 명령문*은 네 가지 방법으로 호출할 수 있습니다.

- 응용프로그램에 포함되어서
- SQL 프로시듀어에서 삽입하여
- 동적으로 준비되어 실행하여서
- 대화식으로 발생하여서

주: REXX에서 삽입된 명령문은 동적으로 준비되고 실행됩니다.

명령문에 따라, 이 방법 중 일부 또는 전부를 사용할 수 있습니다. 각 명령문의 설명에서 호출 절은 사용될 수 있는 방법을 알려줍니다.

*실행 불가능한 명령문*은 응용프로그램에 포함할 수 있습니다.

이 장에 설명된 명령문 외에도, 두 개 이상의 SQL문 구조(*select*문)가 있습니다(482 페이지의 『select문』에서 참조하십시오.). 다른 명령문과 달리 사용되므로, 이 장에 포함되지 않습니다.

*select*문은 세 가지 방법으로 호출될 수 있습니다.

- DECLARE CURSOR에 포함시키고 OPEN, FETCH 및 CLOSE를 통해 내재적으로 수행합니다.
- 동적으로 준비하여 DECLARE CURSOR에서 언급한 다음 OPEN, FETCH 및 CLOSE를 통해 내재적으로 수행합니다.

- 대화식으로 발생하여 수행합니다.

처음의 두 방법을 각각 *select*문의 *정적* 그리고 *동적* 호출이라고 합니다.

SQL문을 호출하는 다른 방법은 아래에서 자세히 설명됩니다. 각 방법에 대해, 설명에는 실행 메카니즘, 호스트 변수를 사용한 상호작용, 그리고 실행 성공 여부에 대한 테스트가 포함됩니다.

응용프로그램에 명령문 포함

SQL문은 사전 처리 컴파일러에 제출될 원시 프로그램에 포함될 수 있습니다. 그러한 명령문을 프로그램에 (*포함*)되었다고 합니다. 포함된 명령문은 호스트 언어 명령문이 허용되는 프로그램의 어느 곳이나 위치될 수 있습니다. 포함된 각 명령문 앞에는 키워드 EXEC 및 SQL이 있어야 합니다.

실행 가능한 명령문

응용프로그램에 삽입된 실행 가능한 명령문은 호스트 언어의 명령문이 같은 위치에 지정된 경우 실행될 때마다 실행됩니다. 그러므로, 루프 내의 명령문은 루프가 실행될 때마다 실행되고, 조건 구조 내의 명령문은 조건이 만족될 때만 실행됩니다.

포함된 명령문에는 호스트 변수에 대한 참조가 포함될 수 있습니다. 이 방법에서 참조되는 호스트 변수는 두 가지 방법으로 사용될 수 있습니다.

- 입력으로(호스트 변수의 현재 값이 명령문 실행시 사용됨)
- 출력으로(명령문 실행의 결과로서 변수에 새 값이 지정됨)

특히, 표현식과 술어에서 호스트 변수에 대한 모든 참조는 변수의 현재 값에 의해 효율적으로 대체됩니다. 즉, 변수는 입력으로 사용됩니다. 다른 참조의 처리는 각 명령문에 대해 개별적으로 설명됩니다.

모든 실행 가능한 명령문 다음에는 SQL 리턴 코드의 테스트가 와야 합니다. 다른 방법으로, WHENEVER문(그 자체가 실행 가능하지 않음)을 사용하여 포함된 명령문의 실행 후에 즉시 제어 흐름을 변경할 수 있습니다.

DML문에서 언급되는 모든 오브젝트는 명령문이 DB2 Universal Database에 바인드될 때 있어야 합니다.

실행 가능하지 않은 명령문

포함된 실행 가능하지 않은 명령문은 사전 처리 컴파일러에 의해서만 처리됩니다. 사전 처리 컴파일러는 명령문에서 발견된 오류를 보고합니다. 이 명령문은 프로그램 실행 중에는 *절대* 처리되지 않습니다. 따라서, 이러한 명령문 다음에는 SQL 리턴 코드가 테스트되어서는 안 됩니다.

SQL 프로시듀어에 명령문 포함

명령문은 CREATE PROCEDURE문의 SQL 프로시듀어 본문 부분에 포함될 수 있습니다. 그러한 명령문을 SQL 프로시듀어에 *포함*되었다고 합니다. SQL 프로시듀어에 삽입될 수 있는 명령문은 1208 페이지의 『SQL 프로시듀어 명령문』에 지정됩니다. 응용프로그램에 삽입된 명령문과는 달리, SQL문 앞에 키워드가 없어도 됩니다. SQL문 설명이 *호스트 변수*를 참조할 때마다, 명령문이 SQL 프로시듀어에 삽입될 때 *SQL 변수*를 사용할 수 있습니다.

동적 준비 및 실행

응용프로그램은 호스트 변수에 위치된 문자열의 양식으로 SQL문을 동적으로 구축합니다. 일반적으로, 명령문은 프로그램에 대해 사용 가능한 일부 데이터(예를 들어, 워크스테이션으로부터의 입력)로부터 구축됩니다. 그렇게 구축된 명령문(select문 제외)은 (포함된) 명령문 PREPARE에 의해 실행을 준비하여 (포함된) 명령문 EXECUTE에 의해 실행될 수 있습니다. 또한, (포함된) 명령문 EXECUTE IMMEDIATE는 한 단계로 명령문을 준비하고 실행할 수 있습니다.

동적으로 준비되는 명령문에는 호스트 변수에 대한 참조사항이 포함되어야 합니다. 대신 매개변수 표시문자를 포함할 수도 있습니다(매개변수 표시문자 규칙에 대해서는 1087 페이지의 『PREPARE』에서 참조하십시오.). 준비된 명령문이 실행되면, 매개변수 표시문자는 EXECUTE문에 지정된 호스트 변수의 현재 값으로 대체됩니다(이 대체사항의 규칙에 대해서는 1016 페이지의 『EXECUTE』에서 참조하십시오.). 일단 준비되면, 명령문은 호스트 변수의 다른 값에 대해 몇번씩 반복할 수 있습니다. 매개변수 표시문자는 EXECUTE IMMEDIATE에서 허용되지 않습니다.

명령문의 성공적이거나 실패한 실행은 EXECUTE(또는 EXECUTE IMMEDIATE) 명령문 다음에 있는 SQL 리턴 코드의 설정값에 의해 표시됩니다. SQL 리턴 코드는 위에 설명된 대로 점검해야 합니다. 505 페이지의 『SQL 리턴 코드』에서 좀 더 자세한 정보를 참조하십시오.

select문의 정적 호출

*select*문은 (실행 가능하지 않은) 명령문 DECLARE CURSOR의 부분으로 포함될 수 있습니다. 그러한 명령문은 (포함된) 명령문 OPEN에 의해 열릴 때마다 실행됩니다. 커서가 열리고 나면, 결과 테이블은 FETCH문의 성공적 실행에 의해 한번에 하나씩 검색될 수 있습니다.

이러한 방법으로 사용되면, *select*문에는 호스트 변수에 대한 참조가 포함할 수 있습니다. 이 참조는 변수가 OPEN 실행시 갖는 값으로 대체됩니다.

select문의 동적 호출

응용프로그램은 호스트 변수에 위치된 문자열의 양식으로 *select*문을 동적으로 구축합니다. 일반적으로, 명령문은 프로그램에 대해 사용 가능한 일부 데이터(예를 들어, 워크스테이션으로부터 확보된 조회)로부터 구축됩니다. 그렇게 구성된 명령문은 (포함된) 명령문 PREPARE에 의해 실행을 위해 준비될 수 있고 (실행 가능하지 않은) 명령문 DECLARE CURSOR에 의해 실행될 수 있습니다. 그러한 명령문은 (포함된) 명령문 OPEN에 의해 커서가 열릴 때마다 실행됩니다. 커서가 열리고 나면, 결과 테이블은 FETCH문의 성공적 실행에 의해 한번에 하나씩 검색될 수 있습니다.

이러한 방법으로 사용되면, *select*문에는 호스트 변수에 대한 참조가 포함되어야 합니다. 대신 매개변수 표시문자를 포함할 수도 있습니다(매개변수 표시문자 규칙에 대해서는 1087 페이지의 『PREPARE』에서 참조하십시오.). 매개변수 표시문자는 OPEN문에 지정된 호스트 변수의 값에 의해 효율적으로 대체됩니다(이 대체에 대한 규칙에 대해서는 1081 페이지의 『OPEN』에서 참조하십시오.).

대화식 호출

워크스테이션에서 SQL문을 입력하는 것은 데이터베이스 관리 프로그램 아키텍처의 일부입니다. 이 방법으로 입력된 명령문은 대화식으로 발행됩니다.

대화식으로 발행된 명령문은 매개변수 표시문자나 호스트 변수에 대한 참조사항을 포함하지 않는 실행 가능한 명령문이어야 합니다. 이 표시문자는 응용프로그램의 문맥에서만 감지되기 때문입니다.

SQL 리턴 코드

실행 가능한 SQL문을 포함하는 응용프로그램은 SQLCODE 또는 SQLSTATE 값을 사용하여 SQL문의 리턴 코드를 처리할 수 있습니다. 다음의 두 방법으로 응용프로그램은 이들 값에 액세스할 수 있습니다.

- SQLCA 구조를 포함시킵니다. SQLCA는 REXX에서 자동으로 제공됩니다. 다른 언어에서, SQLCA는 INCLUDE SQLCA문을 사용하여 확보할 수 있습니다.

  SQLCA에는 SQLCODE라고 하는 정수 변수와 SQLSTATE라고 하는 문자열 변수가 포함됩니다.

- LANGLEVEL SQL92E가 사전 처리 컴파일 옵션으로 지정되는 경우, 변수 SQLCODE 또는 SQLSTATE가 프로그램의 SQL 선언 섹션에 선언됩니다. 이들 변수 중 어떤 것도 SQL 선언 섹션에 선언되지 않을 경우, 변수 SQLCODE가 프로그램의 임의의 지점에서 선언된 것으로 간주됩니다. LANGLEVEL SQL92E를 사용할 경우, 프로그램에 INCLUDE SQLCA문이 있으면 안 됩니다.

가끔, 경고 조건이 리턴 코드에 따라 오류 조건에 추가로 나오게 됩니다. 경고 SQLCODE는 양의 값이고, 경고 SQLSTATE의 처음 두 문자는 '01'로 설정됩니다.

SQLCODE

SQLCODE는 각 SQL문이 실행된 후에 데이터베이스 관리 프로그램에 의해 설정됩니다. 모든 데이터베이스 관리 프로그램은 다음과 같이 ISO/ANSI SQL 표준에 따릅니다.

- SQLCODE = 0 및 SQLWARN0가 공백이면, 실행은 성공한 것입니다.
- SQLCODE = 100이면, "데이터가 전혀" 없었습니다. 예를 들어, FETCH문은 커서가 결과 테이블의 마지막 행 뒤에 위치되었기 때문에 어떤 데이터도 리턴하지 않았습니다.
- SQLCODE가 0보다 크고 100과 같지 않을 경우, 실행은 성공하지만 경고가 표시됩니다.

- SQLCODE = 0이고 SQLWARN0 = 'W'이면, 실행이 성공적이었으나, 하나 이상의 경고 표시기가 설정된 것입니다. 1261 페이지의 『부록B. SQL 통신 (SQLCA)』에서 자세한 내용을 참조하십시오.
- SQLCODE가 0보다 작을 경우, 실행에 실패하였습니다.

0과 100이 아닌 SQLCODE 값의 의미는 제품마다 다릅니다. 제품별 의미에 대해서는 *메시지 참조서*에서 참조하십시오.

SQLSTATE

SQLSTATE도 각 SQL문이 실행된 후에 데이터베이스 관리 프로그램에 의해 설정됩니다. 그러므로, 응용프로그램은 SQLCODE 대신 SQLSTATE를 테스트하여 SQL문의 실행을 점검할 수 있습니다.

SQLSTATE는 공통 오류 조건에 대해 공통 코드를 갖고 있는 응용프로그램을 제공합니다. 또한, SQLSTATE는 응용프로그램이 특정 오류나 오류 클래스에 대해 테스트될 수 있도록 설계되어 있습니다. 코딩 계획은 모든 IBM 데이터베이스 관리 프로그램에 대해 동일하며 ISO/ANSI SQL92 표준에 따라 달라집니다. SQLSTATE에 대해 가능한 값의 전체 목록을 보려면 *메시지 참조서*에서 참조하십시오.

SQL 주석

정적 SQL문에는 호스트 언어나 SQL 주석을 포함할 수 없습니다. SQL 주석은 두 개의 하이픈으로 표시됩니다.

이 규칙은 SQL 주석 사용에 적용됩니다.

- 두 개의 하이픈은 같은 행에 있어야 하고, 공백으로 구분되지 않습니다.
- 주석은 공간이 유효할 때 어디에서나 시작할 수 있습니다(분리문자 토큰 내에서나 'EXEC' 및 'sQL' 사이에서는 시작할 수 없습니다).
- 주석은 행의 끝(EOL)에 의해 종료됩니다.
- 주석은 동적으로 준비되는(PREPARE 또는 EXECUTE IMMEDIATE를 사용하여) 명령문 내에서 허용되지 않습니다.
- COBOL에서, 하이픈 앞에 공백이 있어야 합니다.

예: 이 예는 C 프로그램 내에서 SQL문에 주석을 포함시키는 방식을 보여줍니다.

```
EXEC SQL
  CREATE VIEW PRJ_MAXPER          -- projects with most support personnel
    AS SELECT PROJNO, PROJNAME    -- number and name of project
         FROM PROJECT
         WHEREDEPTNO = 'E21'     -- systems support dept code
         AND PRSTAFF > 1;
```

ALTER BUFFERPOOL

ALTER BUFFERPOOL문은 다음을 수행할 때 사용됩니다.

- 모든 파티션(또는 노드) 또는 단일 파티션상의 버퍼 풀 *크기* 수정
- 확장 기억영역의 사용을 허가 또는 금지
- 버퍼 풀 정의를 새로운 노드 그룹에 추가

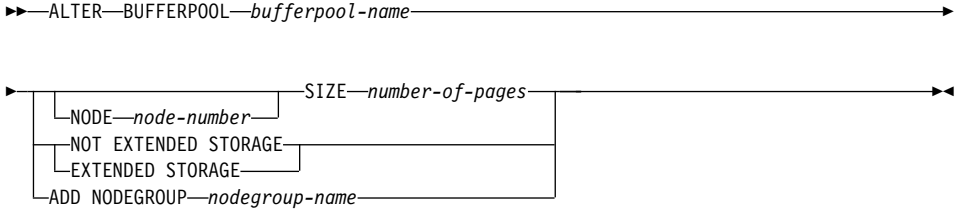
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID는 SYSCTRL 또는 SYSADM 권한이 있어야 합니다.

구문



설명

버퍼풀 이름

버퍼 풀에 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. 이것은 SQL 식별자(일반 또는 분리 식별자)입니다. 이것은 카탈로그에 기술된 버퍼 풀이어야 합니다.

NODE *노드 번호*

버퍼 풀 *크기*가 수정되는 파티션을 지정합니다. 파티션은 버퍼 풀에 대한 노드 그룹 가운데 하나여야 합니다(SQLSTATE 42729). 이 절이 지정되지 않을 경

우, 버퍼 풀(CREATE 버퍼 풀 명령문의 *except-on-nodes*절에 지정된 *크기*가 아닌)에 대해 *기본 크기*를 사용한 버퍼 풀이 있는 모든 파티션에서 버퍼 풀의 *크기*가 수정됩니다.

SIZE *페이지 수*

페이지 수로 지정되는 버퍼 풀의 *크기*. [56]

EXTENDED STORAGE

확장 기억영역 구성이 on인 경우, [57]이 버퍼 풀에서 이주되는 페이지는 확장 저장영역에 캐쉬됩니다.

NOT EXTENDED STORAGE

확장 기억영역 구성이 on인 경우라도, 이 버퍼 풀로부터 이주되고 있는 페이지가 확장 기억영역에 캐쉬되지 않습니다.

ADD NODEGROUP *노드 그룹 이름*

버퍼 풀 정의를 적용할 수 있는 노드 그룹 목록에 이 노드 그룹을 추가합니다. 아직 버퍼 풀이 정의되지 않은 노드 그룹에 있는 파티션의 경우, 버퍼 풀은 버퍼 풀에 대해 지정된 *기본 크기*를 사용하여 파티션에서 작성됩니다. *노드 그룹 이름*의 테이블 공간이 이 버퍼 풀을 지정할 수도 있습니다. 노드 그룹이 데이터베이스에 존재해야 합니다(SQLSTATE 42704).

주

- 버퍼 풀 정의가 트랜잭션 가능하며, 버퍼 풀 정의에 대한 변경이 확약시 카탈로그 테이블에 반영된다고 하여도, 다음 번에 데이터베이스가 시작될 때까지 실질적인 버퍼 풀에 대한 변경은 효력을 갖지 않습니다. 버퍼 풀에 대한 현재 속성은 그 때까지 존재하며, 그 사이에는 버퍼 풀에 어떤 영향도 가지 않습니다. 새로운 노드 그룹의 테이블 공간에 작성된 테이블은 버퍼 풀 기본값을 사용합니다.

---

56. (-1)의 값으로 *크기*를 지정할 수 있는 데, 이 값은 BUFFPAGE 데이터베이스 구성 매개변수의 버퍼 풀 *크기*를 취해야 함을 나타냅니다.

57. 확장 기억영역 구성은 데이터베이스 구성 매개변수 NUM_ESTORE_SEGS와 ESTRORE_SEG_SIZE를 0이 아닌 값으로 설정하면 on으로 됩니다. *관리 안내서*에서 자세한 내용을 참조하십시오.

- 데이터베이스 관리 프로그램과 응용프로그램의 나머지 부분에 대해서뿐 아니라 전체 버퍼 풀에 대해 머신상에 충분한 실제 메모리가 있어야 합니다.

ALTER NICKNAME

ALTER NICKNAME문은 다음과 같이 함으로써 데이터 소스 테이블이나 뷰의 연합 데이터베이스 표현을 수정합니다.

- 테이블이나 뷰 컬럼의 지역 이름을 변경
- 이들 컬럼의 지역 데이터 유형을 변경
- 이들 컬럼에 대한 옵션을 추가, 변경 또는 삭제

호출

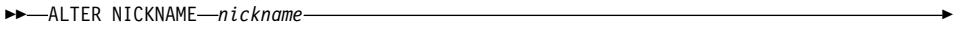
이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

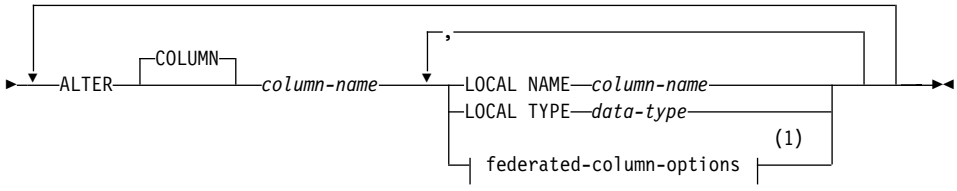
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

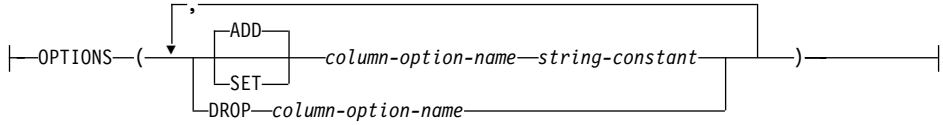
- SYSADM 또는 DBADM 권한
- 명령문에서 지정된 별명(nickname)에 대한 ALTER 특권
- 명령문에서 지정된 별명(nickname)에 대한 CONTROL 특권
- 별명의 스키마 이름이 존재하는 경우에는, 스키마상의 ALTERIN 특권
- 별명에 대한 카탈로그 뷰의 DEFINER 컬럼에 기록된 대로 별명의 정의자

구문





federated-column-options:



주:

1 사용자가 LOCAL NAME 매개변수나 LOCAL TYPE 매개변수 또는 이들 매개변수 둘다를 지정하는 것외에, federated-column-options절을 지정해야 하는 경우에는, federated-column-options절을 마지막에 지정해야 합니다.

설명

별명(nickname)

COLUMN 키워드 다음에 지정된 컬럼을 포함하는 데이터 소스 테이블이나 뷰에 대한 별명을 식별합니다. 카탈로그에 기술된 별명이어야 합니다.

ALTER COLUMN *컬럼 이름*

변경될 컬럼에 이름을 부여합니다. *column-name*은 데이터 소스에 있는 테이블이나 뷰의 컬럼에 대한 연합 서버의 현재 이름입니다. *column-name*은 *별명*에 의해 참조되는 데이터 소스 테이블이나 뷰의 기존 컬럼을 식별해야 합니다.

LOCAL NAME *컬럼 이름*

연합 서버가 ALTER COLUMN *column-name* 매개변수에 의해 식별된 컬럼을 참조할 새로운 이름입니다. 이 새로운 이름은 유효한 DB2 식별자여야 합니다.

LOCAL TYPE *데이터 유형*

지정된 컬럼의 데이터 유형을 새로운 것에 맵핑하는 것이 아닌 다른 지역 데이터 유형에 맵핑합니다. 새로운 유형은 *data-type*으로 표시됩니다.

*data-type*은 LONG VARCHAR, LONG VARGRAPHIC, DATALINK, 대형 오브젝트(LOB) 데이터 유형 또는 사용자 정의 유형일 수 없습니다.

OPTIONS

COLUMN 키워드 뒤에 지정된 컬럼에 대해 어느 컬럼 옵션이 작동 가능, 재설정 또는 삭제될지를 나타냅니다. *컬럼 옵션 이름* 및 그 설정값에 대한 설명은 1405 페이지의 『컬럼 옵션』에서 참조하십시오.

ADD

컬럼 옵션을 작동 가능하게 합니다.

SET

컬럼 옵션의 설정값을 변경합니다.

컬럼 옵션 이름

작동 가능화되거나 재설정될 컬럼 옵션에 이름을 부여합니다.

문자열 상수

*column-option-name*에 대한 설정값을 문자열 상수로서 지정합니다.

DROP *컬럼 옵션 이름*

컬럼 옵션을 삭제합니다.

규칙

- 뷰가 별명(nickname)에서 작성되었으면, 별명이 참조하는 테이블이나 뷰에 있는 컬럼에 대한 지역 이름이나 데이터 유형을 변경하는 데 ALTER NICKNAME문을 사용할 수 없습니다(SQLSTATE 42601). 그러나, 이들 컬럼에 대한 컬럼 옵션을 작동 가능, 재설정 또는 삭제하는 데 이 명령문을 사용할 수 있습니다.

주

- ALTER NICKNAME을 사용하여 별명이 참조하는 테이블이나 뷰에 있는 컬럼에 대한 지역 이름을 변경하면, 컬럼의 조회가 새로운 이름으로 그것을 참조해야 합니다.

- 동일한 ALTER NICKNAME문에서 컬럼 옵션이 두번 이상 지정될 수는 없습니다(SQLSTATE 42853). 컬럼 옵션이 작동가능화, 재설정 또는 삭제되면, 사용중인 모든 다른 컬럼 옵션들에 영향이 미치지 않습니다.
- 컬럼 데이터 유형의 지역 스펙이 변경될 때, 데이터베이스 관리 프로그램은 그 컬럼에 대해 수집된 어떠한 통계든(HIGH2KEY, LOW2KEY 등등) 무효화시킵니다.
- 이 명령문에서 참조된 별명(nickname)이 이미 동일한 작업 단위(UOW)에서 SELECT문에 의해 참조되는 경우에는, 작업 단위(UOW) 내에서 ALTER NICKNAME문이 처리될 수 없습니다.

예

예 1: 별명 NICK1이 T1이라고 하는 AS/400용 DB2 Universal Database 테이블을 참조하십시오. 또한, COL1은 이 테이블의 첫번째 컬럼인 C1을 참조하는 지역 이름입니다. C1에 대한 지역 이름을 NEWCOL로 변경하십시오.

```
ALTER NICKNAME NICK1
   ALTER COLUMN COL1
   LOCAL NAME NEWCOL
```

예 2: 별명 EMPLOYEE가 EMP라고 하는 OS/390용 DB2 Universal Database 테이블을 참조합니다. 또한, SALARY는 이 테이블의 컬럼 중 하나인 EMP_SAL을 참조하는 지역 이름입니다. 컬럼의 데이터 유형 FLOAT가 지역 데이터 유형 DOUBLE로 맵핑됩니다. FLOAT가 DECIMAL (10, 5)로 맵핑되도록 맵핑을 변경하십시오.

```
ALTER NICKNAME EMPLOYEE
   ALTER COLUMN SALARY
   LOCAL TYPE DECIMAL(10,5)
```

예 3: Oracle 테이블에서 VARCHAR의 데이터 유형을 가진 컬럼이 뒤 공백을 가지지 않는다고 나타내십시오. 테이블의 별명은 NICK2이며, 컬럼의 지역 이름은 COL1입니다.

```
ALTER NICKNAME NICK2
   ALTER COLUMN COL1
   OPTIONS ( ADD VARCHAR_NO_TRAILING_BLANKS 'Y' )
```

ALTER NODEGROUP

ALTER NODEGROUP문은 다음을 수행하는 데 사용됩니다.

- 하나 이상의 파티션 또는 노드를 노드 그룹에 추가
- 하나 이상의 파티션을 노드 그룹에서 삭제

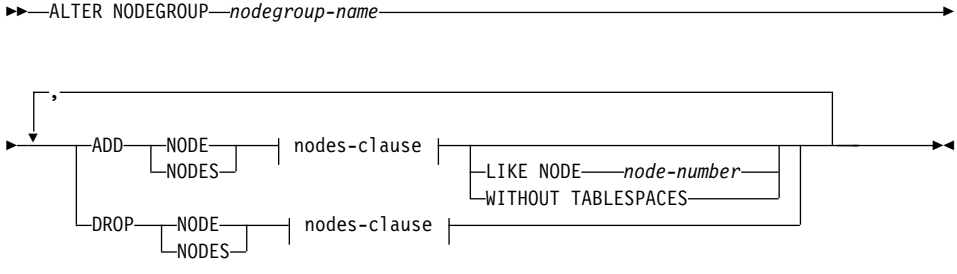
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

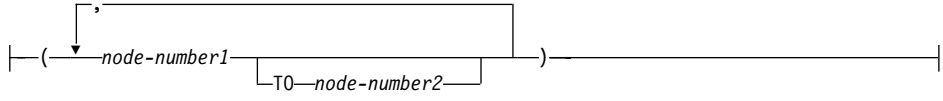
권한 부여

명령문의 권한 부여 ID는 SYSCTRL 또는 SYSADM 권한이 있어야 합니다.

구문



nodes-clause:



설명

노드 그룹 이름

노드 그룹의 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. 이것은 SQL

식별자(일반 또는 분리 식별자)입니다. 이것은 카탈로그에 기술된 노드 그룹이어야 합니다. IBMCATGROUP와 IBMTEMPGROUP는 지정할 수 없습니다(SQLSTATE 42832).

ADD NODE
노드 그룹에 추가할 특정 파티션이나 파티션들을 지정합니다. NODES는 NODE와 동의어입니다. 지정되는 파티션은 노드 그룹에 정의되어 있지 않아야 합니다(SQLSTATE 42728).

DROP NODE
노드 그룹에서 삭제할 특정 파티션이나 파티션들을 지정합니다. NODES는 NODE와 동의어입니다. 지정된 파티션은 노드 그룹에 정의되어 있지 않아야 합니다(SQLSTATE 42729).

노드 절
추가하거나 삭제할 파티션이나 파티션들을 지정합니다.

노드 번호1
특정 파티션 번호를 지정합니다.

TO *노드 번호2*
파티션 번호의 범위를 지정합니다. 노드 *번호2*의 값은 노드 *번호1*의 값보다 크거나 같아야 합니다(SQLSTATE 428A9).

LIKE NODE *노드 번호*
노드 그룹에 있는 기존 테이블 공간에 대한 컨테이너가 지정된 노드 번호의 컨테이너와 같도록 지정합니다. 지정되는 파티션은 이 명령문 이전에 노드 그룹에 있던 파티션이어야 하며 같은 명령문의 DROP NODE절에는 포함되지 않아야 합니다.

WITHOUT TABLESPACES
새로 추가된 파티션이나 파티션들에 기본 테이블 공간이 작성되지 않도록 지정합니다. 이 노드 그룹에 정의된 테이블 공간에 사용할 컨테이너를 정의하려면 FOR NODE절을 사용하는 ALTER TABLESPACE를 사용해야 합니다. 이 옵션이 지정되지 않을 경우, 노드 그룹상에 정의된 각 테이블 공간에 대해 새로 추가된 파티션에 기본 컨테이너가 지정됩니다.

규칙

- 번호별로 지정되는 파티션이나 노드는 db2nodes.cfg 파일에 정의되어야 합니다(SQLSTATE 42729). 이 파일에 대해서는 68 페이지의 『여러 파티션에 걸친 데이터 파티션』에서 자세한 내용을 참조하십시오.
- ON NODES절에 나열된 각 노드 번호는 고유한 파티션에 대한 번호여야 합니다(SQLSTATE 42728).
- 유효한 파티션 번호는 0 - 999 사이입니다(SQLSTATE 42729).
- 파티션은 ADD 및 DROP 절 둘 다에서 표시될 수 없습니다(SQLSTATE 42728).
- 노드 그룹에 최소한 하나의 파티션이 남아 있어야 합니다. 마지막 파티션은 노드 그룹에서 삭제될 수 없습니다(SQLSTATE 428C0).
- 파티션을 추가할 때 LIKE NODE 절이나 WITHOUT TABLESPACES 절을 지정하지 않은 경우, 기본값은 노드 그룹에서 기존 파티션의 최하위 파티션 번호(2라고 하고)를 사용하여 LIKE NODE 2가 지정된 것처럼 진행하는 것입니다. 기본값으로 사용될 기존 파티션의 경우, 노드 그룹의 모든 테이블 공간에 대해 컨테이너가 정의되어야 합니다(SYSCAT.NODEGROUPDEF의 컬럼 IN_USE가 'T'가 아님).

주

- 파티션이나 노드가 노드 그룹에 추가될 경우, 파티션에 대한 카탈로그 항목이 입력됩니다(SYSCAT.NODEGROUPDEF 참조). 파티션 맵은 파티션이 다음 중 어느 한 경우에 해당될 때 파티션이 파티션 맵에 있음을 나타내는 표시기(IN_USE)와 함께 새 파티션을 포함시키기 위해 바로 변경됩니다.
  - 노드 그룹에 테이블 공간이 정의되지 않은 경우
  - 노드 그룹에 정의된 테이블 공간에 테이블이 정의되지 않고 WITHOUT TABLESPACES절이 지정되지 않은 경우.

  파티션 맵은 변경되지 않고 표시기(IN_USE)는 다음 중 어느 한 경우에 해당될 때 파티션이 파티션 맵에 포함되어 있지 않음을 나타내도록 설정됩니다.
  - 테이블이 노드 그룹의 테이블 공간에 존재합니다.

– 테이블 공간이 노드 그룹에 존재하고 WITHOUT TABLESPACES절이 지정되었습니다.

파티션 맵핑을 변경하려면 REDISTRIBUTE NODEGROUP 명령을 사용해야 합니다. 이것은 데이터를 다시 분산시키고 파티션 맵을 변경한 후 표시기를 변경합니다. 테이블 공간 컨테이너는 WITHOUT TABLESPACES 절을 지정하지 않은 경우에 데이터 재분배를 시도하기 전에 추가되어야 합니다.

- 파티션이 노드 그룹에서 삭제될 경우, 파티션에 대해 카탈로그 항목이 갱신됩니다(SYSCAT.NODEGROUPDEF 참조). 노드 그룹에 정의된 테이블 공간에 테이블이 정의되지 않은 경우, 노드 그룹의 파티션에 대한 항목과 삭제된 파티션을 바로 제외시키기 위해 파티션 맵이 변경됩니다. 테이블이 존재할 경우, 파티션 맵은 변경되지 않고 표시기(IN_USE)는 파티션이 삭제되기를 기다리고 있음을 표시하도록 설정됩니다. REDISTRIBUTE NODEGROUP 명령을 사용하여 데이터를 다시 분산시키고 파티션에 대한 항목을 노드 그룹에서 삭제해야 합니다.

예

파티션 0, 1, 2, 5, 7, 8이 있는 6 파티션 데이터베이스가 있다고 가정합시다. 두 파티션이 파티션 번호 3과 6으로 시스템에 추가되었습니다.

- 파티션이나 노드 3과 6을 MAXGROUP을 호출한 노드 그룹에 추가하려 하고 그와 같은 테이블 공간 컨테이너가 파티션 2에 있다고 가정합시다. 명령문은 다음과 같습니다.

```
ALTER  NODEGROUP MAXGROUP
ADD NODES (3,6) LIKE NODE 2
```

- 파티션 1을 삭제하고 파티션 6을 노드 그룹 MEDGROUP에 추가하려 한다고 가정합시다. ALTER TABLESPACE를 사용하여 파티션 6에 대해 별도의 테이블 공간 컨테이너를 정의하게 됩니다. 명령문은 다음과 같습니다.

```
ALTER NODEGROUP MEDGROUP
 ADD NODE(6) WITHOUT TABLESPACES
 DROP NODE(1)
```

ALTER SERVER

ALTER SERVER문은[58] 다음을 위해 사용됩니다.

- 특정 데이터 소스의 정의나 데이터 소스의 카테고리의 정의를 수정하기 위해.
- 특정 데이터 소스의 구성이나, 연합 데이터베이스에 대한 여러 연결에서 지속될 데이터 소스 변경사항의 범주 구성에서의 변경을 위해.

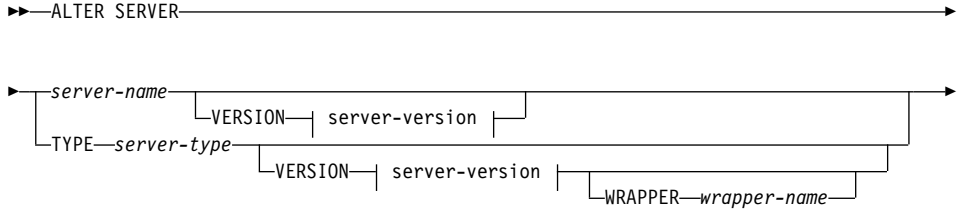
호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

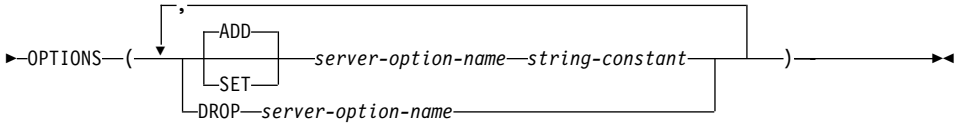
권한 부여

명령문의 권한 부여 ID는 연합 데이터베이스상의 SYSADM 또는 DBADM 권한을 포함해야 합니다.

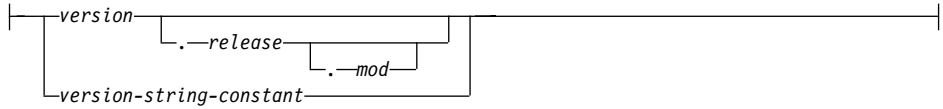
구문



58. 이 명령문에서, SERVER라는 단어와 *server*-로 시작하는 매개변수 이름은 연합 시스템에 있는 데이터 소스만 지칭합니다. 그러한 시스템에 있는 연합 서버나, DRDA 응용프로그램 서버(AS)는 지칭하지 않습니다. 연합 시스템에 관한 47 페이지의 『DB2 연합 시스템』에서 자세한 정보를 참조하십시오. DRDA 응용프로그램 서버(AS)에 대한 34 페이지의 『분산 관계형 데이터베이스』에서 자세한 정보를 참조하십시오.



server-version:



설명

서버 이름

요청되고 있는 변경사항이 적용할 데이터 소스에 대한 연합 서버의 이름을 식별합니다. 데이터 소스는 카탈로그에서 기술되어 있는 것이어야 합니다.

VERSION

server-name 다음의 VERSION 및 그 매개변수는 *server-name*이 설명하는 데이터 소스의 새로운 버전을 지정합니다.

버전

버전 번호를 지정합니다. *버전*은 정수여야 합니다.

릴리스

*버전*으로 표시된 버전의 릴리스 번호를 지정합니다. *릴리스*는 정수여야 합니다.

mod

*릴리스*로 표시된 릴리스의 수정 번호를 지정합니다. *mod*는 정수여야 합니다.

버전 문자열 상수

완전한 버전 지정을 합니다. *버전 문자열 상수*는 단일 값(예를 들어, '8i') 이거나 또는 *버전*, *릴리스* 및 *mod*의 결합된 값(예를 들어, '8.0.3')일 수 있습니다.

TYPE *서버 유형*

요청되고 있는 변경사항이 적용할 데이터 소스의 유형을 식별합니다. 서버 유형은 카탈로그에서 나열되는 것이어야 합니다.

VERSION

server-type 뒤의, VERSION 및 그 매개변수는 어느 서버 옵션에 대한 데이터 소스의 버전이 작동가능, 재설정 또는 삭제될 것인지를 지정합니다.

WRAPPER *랩퍼 이름*

server-type 및 *server-version*에 의해 표시된 유형 및 버전의 데이터 소스와 상호 작용하기 위해 연합 서버가 사용하는 랩퍼의 이름을 지정합니다. 랩퍼는 카탈로그에서 나열되어야 합니다.

OPTIONS

*server-name*에 의해 표시되는 데이터 소스나, *server-type* 및 그 연관 매개변수에 의해 표시되는 데이터 소스의 카테고리에 대해 어느 서버 옵션을 작동가능, 재설정 또는 삭제할 것인지를 나타냅니다. *서버 옵션 이름* 및 그 설정값에 대한 설명은 1407 페이지의 『서버 옵션』에서 참조하십시오.

ADD

서버 옵션을 작동가능하게 합니다.

SET

서버 옵션의 설정값을 변경합니다.

서버 옵션 이름

작동가능화되거나 재설정될 서버 옵션에 이름을 부여합니다.

문자열 상수

*server-option-name*에 대한 설정값을 문자열 상수로서 지정합니다.

DROP *서버 옵션 이름*

서버 옵션을 삭제합니다.

주

- 이 명령문은 DBNAME 및 NODE 서버 옵션을 지원하지 않습니다(SQLSTATE 428EE).

- 서버 옵션은 동일한 ALTER SERVER문에서 두번 이상 지정될 수 없습니다 (SQLSTATE 42853). 서버 옵션이 작동가능화, 재설정 또는 삭제될 때, 사용 중인 모든 다른 서버 옵션들에는 영향이 미치지 않습니다.
- 주어진 작업 단위(UOW) 내의 ALTER SERVER문은 다음 조건에서는 처리될 수 없습니다.
  - 명령문이 단일 데이터 소스를 참조하며, UOW가 이미 이 데이터 소스 내의 테이블이나 뷰에 대한 별명(nickname)을 참조하는 SELECT문을 포함합니다(SQLSTATE 55007).
  - 명령문이 데이터 소스의 카테고리를 참조하며(예를 들어, 특정 유형 및 버전의 모든 데이터 소스), UOW가 이미 이들 데이터 소스 중 하나 안에 있는 테이블이나 뷰에 대한 별명(nickname)을 참조하는 SELECT문을 포함합니다(SQLSTATE 55007).
- 서버 옵션이 데이터 소스의 한 유형에 대해 어느 한 값으로 설정되고, 유형의 한 인스턴스에 대해 또 다른 값으로 설정되면, 인스턴스를 위해 두 번째 값이 첫번째 값을 겹쳐씁니다. 예를 들어, 서버 유형 ORACLE에 대해 PLAN_HINTS가 'Y'로 설정되고, DELPHI라는 이름의 Oracle 데이터 소스에 대해 'N'으로 설정된다고 가정하십시오. 이 구성은 DELPHI를 제외한 모든 Oracle 데이터 소스에서 플랜 힌트가 작동가능화되게 합니다.

예

예 1: 권한 부여 ID가 Oracle 8.0.3 데이터 소스로 전송될 때, ID의 대소문자가 변경되지 않게 하십시오. 또한, 이들 데이터 소스가 지역 CPU 보다 절반만큼 빠른 업그레이드된 CPU에서 실행하도록 시작되었다고 가정하십시오. 최적화 알고리즘에게 이 상태를 알리십시오.

```
ALTER SERVER
   TYPE ORACLE
   VERSION 8.0.3
   OPTIONS
      ( ADD FOLD_ID 'N',
      SET CPU_RATIO '2.0' )
```

예 2: SUNDIAL이라고 하는 AS/400용 DB2 Universal Database 버전 3.0 데이터 소스가 버전 3.1로 업그레이드되었다고 가정하십시오.

```
ALTER SERVER SUNDIAL
   VERSION 3.1
```

ALTER TABLE

ALTER TABLE문은 다음과 같이 기존 테이블을 수정합니다.

- 테이블에 하나 이상의 컬럼 추가
- 기본 키 추가 또는 삭제
- 하나 이상의 참조 제한조건 추가 또는 삭제
- 하나 이상의 점검 제한조건 정의 추가 또는 삭제
- VARCHAR 컬럼 길이 변경
- 영역을 추가할 참조 유형 컬럼 변경
- 생성된 컬럼의 생성 표현식 변경
- 파티션 키 추가 또는 삭제
- 데이터 캡쳐 옵션, pctfree, 잠금 크기 또는 부가 모드와 같은 테이블 속성 변경
- 초기에 로그되지 않은 상태로 테이블 설정

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- 변경될 테이블에 대해 ALTER 특권
- 변경될 테이블에 대해 CONTROL 특권
- 테이블의 스키마에 대한 ALTERIN 특권
- SYSADM 또는 DBADM 권한

외부 키를 작성하거나 삭제하려면, 명령문의 권한 부여 ID에서 갖고 있는 특권에는 상위 테이블에 대해 다음 중 하나를 포함해야 합니다.

- 테이블에 대해 REFERENCES 특권
- 지정된 상위 키의 각 컬럼에 대한 REFERENCES 특권
- 테이블에 대해 CONTROL 특권
- SYSADM 또는 DBADM 권한

T 테이블의 고유 제한조건 또는 기본 키를 삭제하려면, 명령문의 권한 부여 ID가 보유한 특권에는 이 T의 상위 키에 종속적인 모든 테이블에 다음 중 적어도 하나가 포함되어야 합니다.

- 테이블에 대해 ALTER 특권
- 테이블에 대해 CONTROL 특권
- 테이블의 스키마에 대한 ALTERIN 특권
- SYSADM 또는 DBADM 권한

요약 테이블이 되도록 테이블을 변경하려면(fullselect를 사용하여), 명령문의 권한 부여 ID에서 보유하고 있는 특권이 최소한 다음 중 하나를 포함해야 합니다.

- 테이블에 대한 CONTROL
- SYSADM 또한 DBADM 권한

그리고, fullselect에서 식별되는 각 테이블이나 뷰에 대해 최소한 다음 중 하나의 특권이 있어야 합니다.

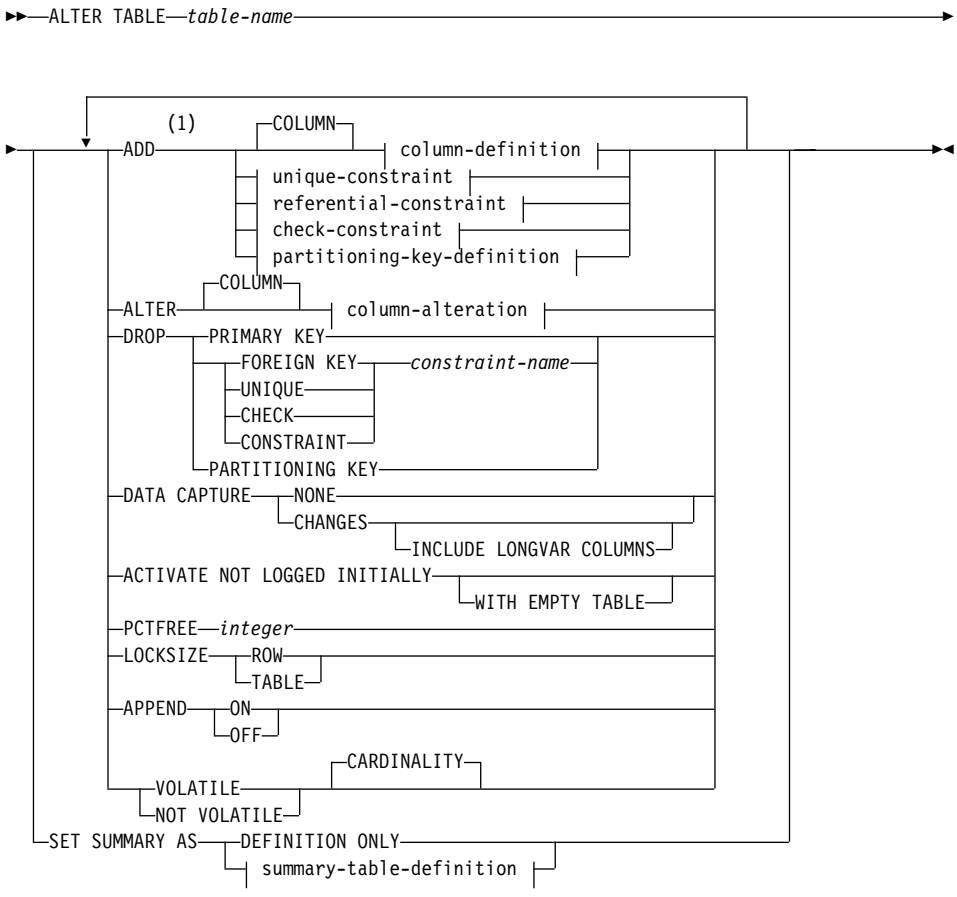
- 테이블 또는 뷰에서의 SELECT 및 ALTER 특권
- 테이블 또는 뷰에서의 CONTROL 특권
- 테이블 또는 뷰에서의 SELECT 특권 및 테이블 또는 특권 스키마에서의 ALTERIN 특권
- SYSADM 또는 DBADM 권한

더이상 요약 테이블이 되지 않도록 테이블을 변경하려면, 요약 테이블을 정의하기 위해 사용되는 fullselect에서 식별되는 각 테이블이나 뷰에 대해, 명령문의 권한 부여 ID가 보유하는 특권에 최소한 다음 중 하나가 포함되어야 합니다.

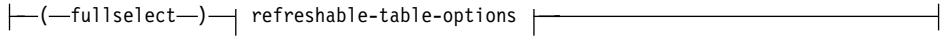
- 테이블 또는 뷰에서의 ALTER 특권
- 테이블 또는 뷰에서의 CONTROL 특권
- 테이블 또는 뷰 스키마에서의 ALTERIN 특권

- SYSADM 또는 DBADM 권한

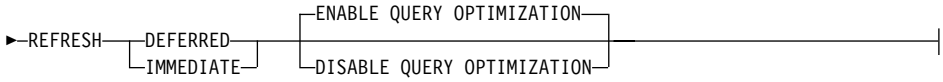
구문



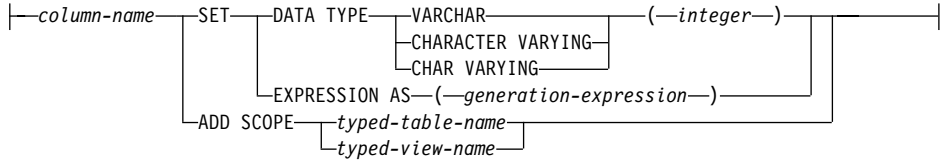
summary-table-definition:



refreshable-table-options:



column-alteration:

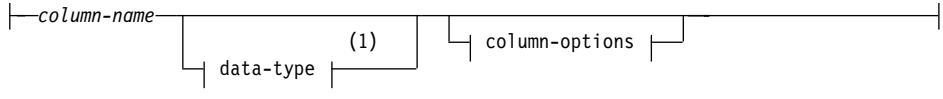


주:

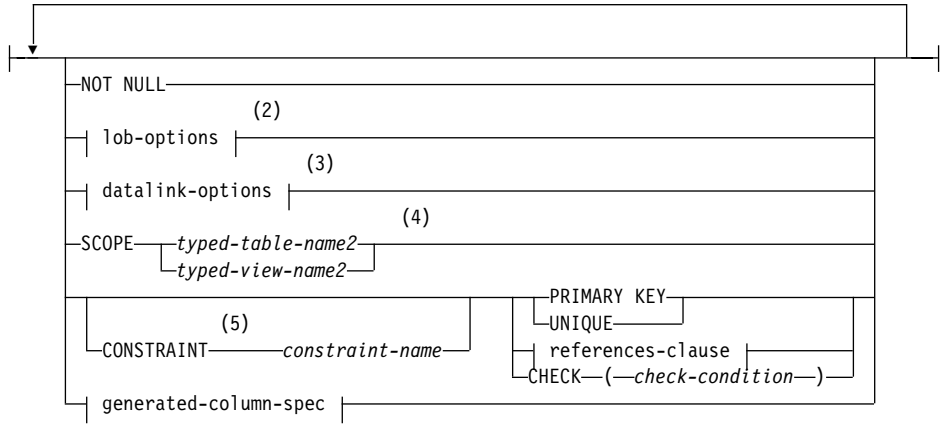
1 버전 1과의 호환성을 위해, 다음에 대해 ADD 키워드는 생략할 수 있습니다.

- 명명되지 않은 PRIMARY KEY 제한조건
- 명명되지 않은 참조 제한조건
- 이름이 FOREIGN KEY 구문에 후속되는 참조 제한조건

column-definition:



column-options:



주:

1 선택된 첫번째 컬럼 옵션이 생성된 컬럼 스펙일 경우, 데이터 유형은 생략하고 생성 표현식에 의해 계산될 수 있습니다.

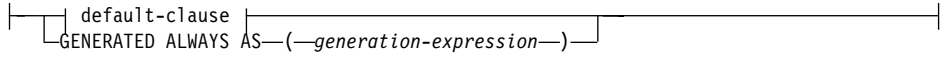
2 lob 옵션 절은 대형 오브젝트 유형(BLOB, CLOB 및 DBCLOB)과 대형 오브젝트(LOB) 유형을 기초로 하는 구별 유형에만 적용됩니다.

3 데이터 링크 옵션 절은 DATALINK 유형과, DATALINK 유형에 기초한 구별 유형에만 적용됩니다.

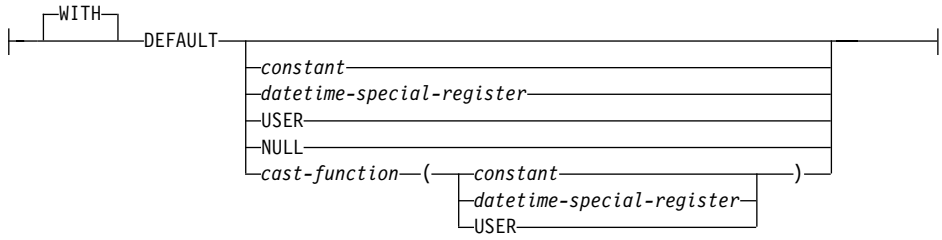
4 SCOPE절은 REF 유형에만 적용됩니다.

5 버전 1과의 호환성을 위해, CONSTRAINT 키워드는 참조 절을 정의하는 *컬럼 정의*에서 생략될 수 있습니다.

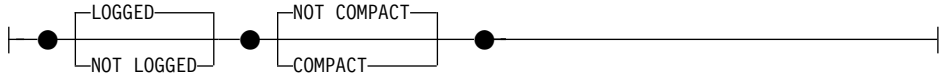
generated-column-spec:



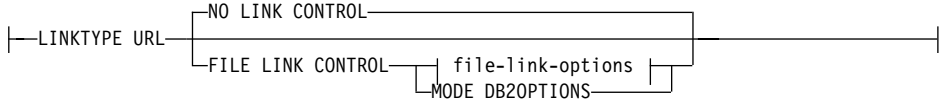
default-clause:



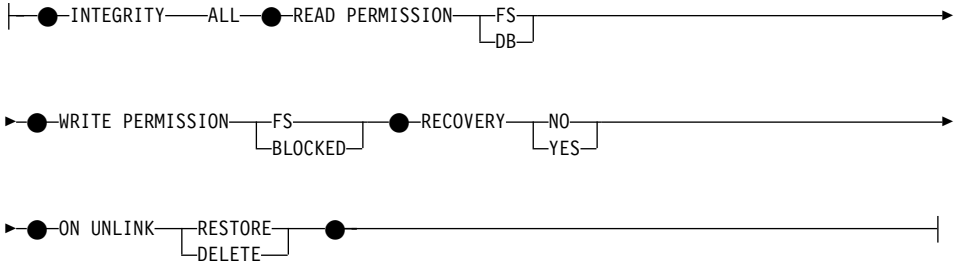
lob-options:



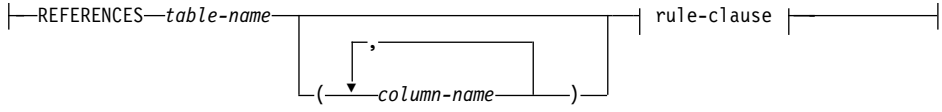
datalink-options:



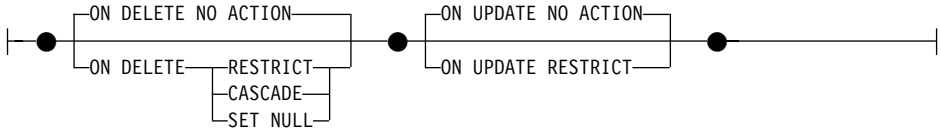
file-link-options:



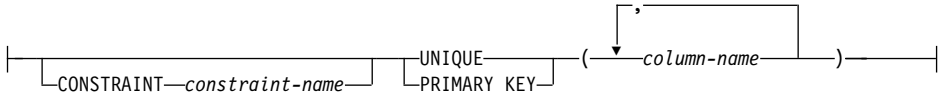
references-clause:



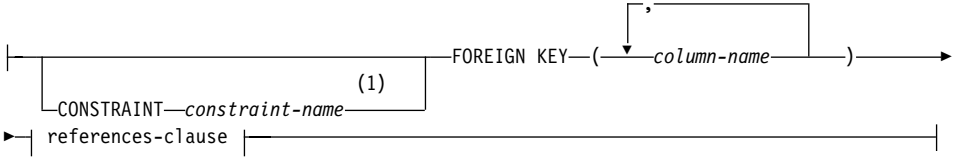
rule-clause:



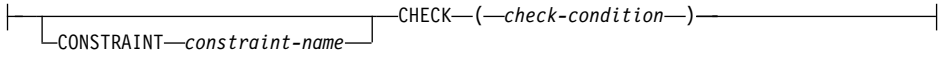
unique-constraint:



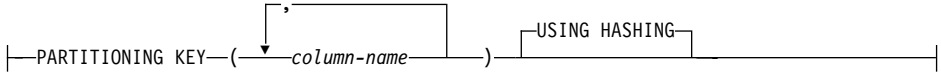
referential-constraint:



check-constraint:



partitioning-key-definition:



주:

1 버전 1과의 호환성을 위해, *제한조건 이름*은 FOREIGN KEY 다음에 지정할 수도 있습니다(CONSTRAINT 키워드 없이).

설명

테이블 이름

변경될 테이블을 식별합니다. 카탈로그에 기술된 테이블이어야 하고 뷰 또는 카탈로그 테이블이어서는 안 됩니다. *테이블 이름*이 요약 테이블을 식별할 경우, 변경사항은 요약 테이블을 정의로만 설정하고, 초기에 활성화가 기록되지 않으

며, pctfree, 잠금 *크기*, 추가 또는 volatile 변경으로 제한됩니다. *테이블 이름*은 별명(SQLSTATE 42809)이나 선언된 임시 테이블(SQLSTATE 42995)이 될 수 없습니다.

SET SUMMARY AS

요약 테이블의 등록 정보에 대한 변경을 허용합니다.

DEFINITION ONLY

더이상 요약 테이블로 간주되지 않도록 요약 테이블을 변경합니다. *테이블 이름*으로 지정된 테이블은 복제되지 않는 요약 테이블로 정의해야 합니다(SQLSTATE 428EW). *테이블 이름*의 컬럼에 대한 정의는 변경되지 않지만, 그 테이블은 조회 최적화에 더이상 사용될 수 없으므로 REFRESH TABLE문을 더이상 사용할 수 없습니다.

summary-table-definition

조회 최적화 동안 사용하기 위해 일반 테이블을 요약 테이블로 변경합니다. *테이블 이름*으로 지정된 테이블은 다음과 같아야 합니다.

- 이전에 요약 테이블로 정의된 적이 없어야 합니다.
- 입력된 테이블이 아니어야 합니다.
- 제한조건, 고유 색인 또는 트리거가 정의되어 있지 않아야 합니다.
- 다른 요약 테이블의 정의에 참조되지 않아야 합니다.

테이블 이름이 이 기준과 일치하지 않을 경우, 오류가 리턴됩니다(SQLSTATE 428EW).

fullselect

테이블이 기본으로 하는 조회를 정의합니다. 기존 테이블의 컬럼은 *fullselect*의 결과 컬럼과 다음 사항이 같아야 합니다(SQLSTATE 428EW).

- 컬럼 수가 같아야 합니다.
- 데이터 유형이 정확하게 같아야 합니다.
- 같은 순서 위치에 같은 컬럼 이름이 있어야 합니다.

요약 테이블에 대해 *fullselect*를 지정하는 방법에 대해서는 800 페이지의 『CREATE TABLE』에서 자세한 내용을 참조하십시오. 하나의 추가 제한사항은 fullselect에서 *테이블 이름*을 직접 또는 간접으로 참조할 수 없다는 것입니다.

refreshable-table-options

요약 테이블 변경을 위해 새로 고칠 수 있는 옵션을 나열합니다.

DATA INITIALLY DEFERRED

테이블의 데이터는 REFRESH TABLE이나 SET INTEGRITY 명령문을 사용하여 유효성이 확인되어야 합니다.

REFRESH

테이블의 데이터가 유지보수되는 방법을 나타냅니다.

DEFERRED

테이블의 데이터가 REFRESH TABLE문을 사용하여 언제라도 화면갱신될 수 있습니다. 테이블의 데이터는 REFRESH TABLE문이 처리될 때의 스냅샷인 조회 결과에만 영향을 미칩니다. 이 속성이 정의되어 있는 요약 테이블에서는 INSERT, UPDATE 또는 DELETE문을 사용할 수 없습니다 (SQLSTATE 42807).

IMMEDIATE

DELETE, INSERT 또는 UPDATE의 일부로서 기저 테이블에 이루어진 변경사항은 요약 테이블에 연쇄됩니다. 이 경우, 특정 시점에 테이블의 내용은 지정된 부속 선택이 처리될 경우와 같습니다. 이 속성이 정의되어 있는 요약 테이블에서는 INSERT, UPDATE 또는 DELETE문을 사용할 수 없습니다 (SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION

요약 테이블이 조회 최적화를 위해 사용될 수 있습니다.

DISABLE QUERY OPTIMIZATION

요약 테이블이 조회 최적화에 사용되지 않습니다. 그 테이블은 여전히 직접 조회됩니다.

ADD *컬럼 정의*

테이블에 컬럼을 추가합니다. 테이블은 입력된 테이블이어서는 안 됩니다 (SQLSTATE 428DH). 테이블에 기존 행이 있을 경우, 새로 추가되는 컬럼의 값이 모두 해당되는 기본값이 됩니다. 새로운 컬럼은 테이블의 마지막 컬럼입니다. 즉, 처음에 *n*개의 컬럼이 있으면, 추가되는 컬럼은 컬럼 *n*+1입니다. *n*의 값은 499보다 클 수 없습니다.

새로운 컬럼을 추가해도 모든 컬럼의 총 바이트 수가 1258 페이지의 표34에서 지정된 최대 레코드 크기를 초과해서는 안 됩니다. 849 페이지의 『주』에서 좀더 자세한 정보를 참조하십시오.

컬럼 이름

테이블에 추가될 컬럼의 이름입니다. 이름을 규정화할 수 없습니다. 테이블에 있는 기존 컬럼 이름은 사용할 수 없습니다(SQLSTATE 42711).

데이터 유형

800 페이지의 『CREATE TABLE』에 나열된 데이터 유형 중 하나입니다.

NOT NULL

컬럼에 널(NULL)값이 포함되지 못하도록 합니다. *기본 절(default-clause)*도 지정해야 합니다(SQLSTATE 42601).

lob 옵션

LOB 데이터 유형에 대해 옵션을 지정합니다. 800 페이지의 『CREATE TABLE』의 *lob 옵션*에서 참조하십시오.

데이터 링크 옵션

DATALINK 데이터 유형에 대한 옵션을 지정합니다. 800 페이지의 『CREATE TABLE』에 있는 *데이터링크 옵션*에서 참조하십시오.

SCOPE

참조 유형 컬럼 영역을 지정합니다.

입력된 테이블 이름2

입력된 테이블의 이름. *컬럼 이름*의 데이터 유형은 REF(*S*)이어야 합니다. 여기서 *S*는 *입력된 테이블 이름2* 유형입니다(SQLSTATE 428DM). 값이 실제로 *입력된 테이블 이름2*의 기존 행을 참조하는지를 확인하기 위해 *column-name*의 기본값을 점검하지는 않습니다.

입력된 뷰 이름2

입력된 뷰의 이름. *컬럼 이름*의 데이터 유형은 REF(*S*)이어야 합니다. 여기서 *S*는 *입력된 뷰 이름2* 유형입니다(SQLSTATE 428DM). 값이 *입력된 뷰 이름2*에 이미 있는 행을 실제로 참조하는지 확인하기 위한 *컬럼 이름*의 기본값에 대한 점검은 수행되지 않습니다.

CONSTRAINT *제한조건 이름*

제한조건을 명명합니다. *제한조건 이름*은 같은 ALTER TABLE문 내에서 또는 테이블에서 임의의 다른 기존의 제한조건 이름으로서 이미 지정된 제한조건과 동일해서는 안 됩니다(SQLSTATE 42710).

사용자가 제한조건 이름을 지정하지 않은 경우, 시스템은 테이블에 정의된 기존 제한조건의 식별자 내에서 고유한 18자의 식별자가 생성됩니다. [59]

PRIMARY KEY 또는 UNIQUE 제한조건과 함께 사용되는 경우, 제한조건을 지원하기 위해 작성된 색인의 이름으로 *제한조건 이름*을 사용할 수 있습니다. 고유성 제한조건과 연관된 색인 이름에 대한 547 페이지의 『주』에서 자세한 내용을 참조하십시오.

PRIMARY KEY

이는 단일 컬럼으로 구성되는 기본 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로, PRIMARY KEY가 컬럼 C의 정의에 지정된 경우, PRIMARY KEY(C)절이 별도의 절로 지정된 경우와 같은 효과를 갖습니다. 컬럼에는 널(NULL) 값이 포함될 수 없으므로, NOT NULL 속성도 지정해야 합니다(SQLSTATE 42831).

아래의 *고유성 제한조건* 설명에 있는 PRIMARY KEY를 참조하십시오.

UNIQUE

이는 단일 컬럼으로 구성되는 고유 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로, UNIQUE KEY가 컬럼 C의 정의에 지정된 경우, UNIQUE KEY(C)절이 별도의 절로 지정된 경우와 같은 효과를 갖습니다.

아래의 *고유 제한조건*의 설명에 있는 UNIQUE를 참조하십시오.

---

59. 식별자는 "SQL"과 그 뒤에 시간소인 함수에 의해 생성된 15개의 연속 숫자로 구성됩니다.

참조 절

이는 단일 컬럼으로 구성되는 외부 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로, 참조 절이 컬럼 C의 정의에 지정되면, C가 유일하게 식별되는 컬럼인 FOREIGN KEY절의 일부로 참조 절이 지정된 것과 동일한 효력을 갖습니다.

800 페이지의 『CREATE TABLE』에서 *참조 절*을 참조하십시오.

CHECK(*점검 조건*)

이는 단일 컬럼에 적용되는 점검 제한조건을 정의함에 있어 간편한 방법을 제공합니다. 800 페이지의 『CREATE TABLE』의 *점검 조건*에서 자세한 내용을 참조하십시오.

generate-column-spec

컬럼 생성에 대해서는 800 페이지의 『CREATE TABLE』에서 자세한 내용을 참조하십시오.

기본 절

컬럼의 기본값을 지정합니다.

WITH

선택적 키워드.

DEFAULT

값이 INSERT에 제공되지 않거나 INSERT 또는 UPDATE의 DEFAULT로 지정된 이벤트에서 기본값을 제공합니다. 특정 기본값이 DEFAULT 키워드 다음에 지정되지 않으면, 기본값은 표19에 표시된 대로 컬럼의 데이터 유형에 따라 달라집니다. 컬럼이 DATALINK나 구조화 유형으로 정의될 경우, DEFAULT 절을 지정할 수 없습니다.

컬럼이 구별 유형을 사용하여 정의되면, 컬럼의 기본값은 구별 유형으로 변환되는 원시 데이터 유형의 기본값입니다.

표 19. 기본값(값이 지정되지 않는 경우)

데이터 유형	기본값
숫자	0
고정 길이 문자열	공백
가변 길이 문자열	길이가 0인 문자열

표 19. 기본값(값이 지정되지 않는 경우) (계속)

데이터 유형	기본값
고정 길이 그래픽 문자열	2바이트 공백
가변 길이 그래픽 문자열	길이가 0인 문자열
날짜	기존의 행의 경우, 1월 1일(0001)에 해당되는 날짜. 추가된 행의 경우, 현재 날짜.
시간	기존의 행의 경우, 0시, 0분, 0초에 해당되는 시간. 추가된 행의 경우, 현재 시간.
시각	기존 행의 경우, 1월 1일(0001)에 해당되는 날짜와, 0시, 0분, 0초에 해당되는 날짜. 추가된 행의 경우, 현재 시각.
2진 문자열(blob)	길이가 0인 문자열

*컬럼 정의*에서 DEFAULT를 생략하면, 컬럼의 기본값으로 널(NULL)값이 사용됩니다.

DEFAULT 키워드로 지정될 수 있는 값의 특정 유형은 다음과 같습니다.

상수

컬럼의 기본값으로 상수를 지정합니다. 지정된 상수는 다음과 같아야 합니다.

- 제3장에 설명된 지정 규칙에 따라 컬럼에 지정할 수 있는 값을 나타냅니다.
- 컬럼이 부동 소수점 데이터 유형으로 정의되어 있지 않는 한, 부동 소수점 상수가 되지 않습니다.
- 상수가 십진수 상수인 경우 컬럼 데이터 유형의 스케일을 넘어 0이 아닌 자릿수를 갖지 않습니다(예를 들면, 1. 234는 DECIMAL(5,2) 컬럼에 대한 기본값이 될 수 있습니다.).
- 따옴표, 16진 상수의 X와 같은 도입 문자 및 상수가 *변환 함수*의 인수일 때 완전한 함수 이름의 문자 및 괄호를 포함하여 254자 이하로 표시됩니다.

날짜시간 특수 레지스터

컬럼의 기본값으로 INSERT 또는 UPDATE시 날짜시간 특수 레지스터 값(CURRENT DATE, CURRENT TIME 또는 CURRENT

TIMESTAMP)을 지정합니다. 컬럼의 데이터 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다 (예를 들면, 데이터 유형은 CURRENT DATE 지정시 DATE여야 함). 기존 행의 경우, 값은 ALTER TABLE문 처리시 현재 날짜, 현재 시간 또는 현재 시각입니다.

USER

컬럼의 기본값으로 INSERT 또는 UPDATE시 USER 특수 레지스터 값을 지정합니다. USER가 지정되면, 컬럼의 데이터 유형은 USER의 길이 속성보다 짧지 않은 길이의 문자열이어야 합니다. 기존 행의 경우, 값은 ALTER TABLE문의 권한 부여 ID입니다.

NULL

컬럼의 기본 값으로 널(NULL)을 지정합니다. NOT NULL이 지정되면, DEFAULT NULL은 동일한 컬럼 정의 내에 지정되어서는 안 됩니다.

변환 함수

이 기본값의 양식은 구별 유형, BLOB 또는 날짜 시간(DATE, TIME 또는 TIMESTAMP) 데이터 유형으로 정의된 컬럼에만 사용될 수 있습니다. 구별 유형의 경우, BLOB 또는 날짜 시간 유형을 따르는 구별 유형을 제외하고는 함수 이름이 컬럼의 구별 유형 이름과 일치해야 합니다. 스키마 이름으로 규정화된 경우, 이는 구별 유형에 대한 스키마 이름과 같아야 합니다. 규정화되지 않았으면, 함수 해석의 스키마 이름은 구별 유형에 대한 스키마 이름과 같아야 합니다. 기본값이 상수인 날짜 시간 유형을 따르는 구별 유형의 경우, 함수가 사용되어야 하며, 함수 이름에서는 구별 유형의 원시 유형과 SYSIBM의 내재적인 또는 명시적인 스키마 이름이 일치해야 합니다. 다른 날짜시간 컬럼의 경우, 해당 날짜 시간 함수가 사용될 수도 있습니다. BLOB나, BLOB를 기초로 하는 구별 유형의 경우, 함수를 사용해야 하고 그 함수의 이름은 SYSIBM의 내재적 또는 명시적 스키마 이름을 가지고 있는 BLOB여야 합니다.

상수

인수로서 상수를 지정합니다. 상수는 구별 유형이 아닌 경우, 구

별 유형의 원시 유형이나 데이터 유형의 상수에 대한 규칙을 따라야 합니다. 변환 함수가 BLOB인 경우, 상수는 리터럴이어야 합니다.

날짜시간 특수 레지스터

CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP를 지정합니다. 컬럼의 구별 유형에 대한 원시 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다.

USER

USER 특수 레지스터. 컬럼의 구별 유형에 대한 원시 유형의 데이터 유형은 최소한 8바이트 길이의 문자열 데이터 유형이어야 합니다. 변환 함수가 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

지정된 값이 유효하지 않으면, 오류(SQLSTATE 42894)가 발생합니다.

ADD *고유 제한조건*

고유성 제한조건이나 기본 키 제한조건을 정의합니다. 서브테이블인 테이블에 기본 키 또는 고유 제한조건을 추가할 수 없습니다(SQLSTATE 429B3). 테이블이 계층 맨 위에 있는 상위 테이블인 경우, 제한조건이 해당 테이블 및 모든 서브테이블에 적용됩니다.

CONSTRAINT *제한조건 이름*

기본 키 또는 고유성 제한조건을 명명합니다. 800 페이지의 『CREATE TABLE』의 *제한조건 이름*에서 자세한 내용을 참조하십시오.

UNIQUE (*컬럼 이름***...,)**

식별된 컬럼들로 구성되는 고유 키를 정의합니다. 식별된 컬럼은 NOT NULL로 정의되어야 합니다. 각 *컬럼 이름*은 테이블의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별되어서는 안 됩니다. 이름을 규정화할 수 없습니다. 식별된 컬럼 수는 16을 초과할 수 없으며 그 컬럼들의 저장 길이 합은 1024를 초과할 수 없습니다(컬럼 저장 길이에 대해서는 853 페이지의 『바이트 계수』에서 자세한 내용을 참조하십시오). 개별적 컬럼의 길이는 255 바이트를 초과할 수 없습니다. LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, 이 유형들 중 하나에 대한 구별 유형 또

는 구조화 유형 컬럼은 고유 키의 부분으로 사용할 수 없습니다(컬럼의 길이 속성이 255 바이트 한계 내에 맞도록 충분히 작은 경우에도)(SQLSTATE 42962). 고유 키의 컬럼 세트는 기본 키나 다른 고유 키의 컬럼 세트와 같을 수 없습니다(SQLSTATE 01543).[60] 식별된 컬럼 세트에 있는 기존의 모든 값은 고유해야 합니다(SQLSTATE 23515).

기존 색인이 고유한 키 정의와 일치하는지를 판별하기 위해 점검이 수행됩니다(색인의 모든 INCLUDE 컬럼은 무시합니다.). 컬럼의 순서나 방향(ASC/DESC) 스펙에 관계없이 동일한 컬럼 세트를 식별할 경우, 색인 정의가 일치합니다. 일치하는 색인 정의가 있는 경우, 시스템에 필요한 것임을 나타내도록 색인 설명이 변경되고, 고유하지 않은 색인일 경우 (고유성을 확인한 후) 고유한 색인으로 변경됩니다. 테이블에 일치하는 색인이 두 개 이상 있을 경우, 기존의 고유 색인이 선택됩니다(선택은 임의적입니다.). 일치하는 색인이 없을 경우, CREATE TABLLE에 기술된 것과 같이 컬럼에 대해 고유성 색인이 자동으로 작성됩니다. 고유성 제한조건과 연관된 색인 이름에 대한 547 페이지의 『주』에서 자세한 내용을 참조하십시오.

PRIMARY KEY *...(컬럼 이름)*

식별된 컬럼들로 구성되는 기본 키를 정의합니다. 각 *컬럼 이름*은 테이블의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별되어서는 안 됩니다. 이름을 규정화할 수 없습니다. 식별된 컬럼 수는 16을 초과할 수 없으며 그 컬럼들의 저장 길이 합은 1024를 초과할 수 없습니다(저장 길이에 대해서는 853 페이지의 『바이트 계수』에서 자세한 내용을 참조하십시오). 개별적 컬럼의 길이는 255 바이트를 초과할 수 없습니다. 테이블에는 기본 키가 없어야 하고 식별된 컬럼은 NOT NULL로 정의되어야 합니다. LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, 이 유형들 중 하나에 대한 구별 유형 또는 구조화 유형 컬럼은 기본 키의 부분으로 사용할 수 없습니다(컬럼의 길이 속성이 255 바이트 한계 내에 맞도록 충분히 작은 경우에도)(SQLSTATE 42962). 기본 키의 컬럼 세트는 고유 키의 컬럼 세트와 같을 수 없습니다(SQLSTATE 01543).[60] 식별된 컬럼 세트의 기존 값은 고유해야 합니다(SQLSTATE 23515).

60. LANGLEVEL이 SQL92E 또는 MIA일 경우, 오류가 리턴됩니다(SQLSTATE 42891).

기존 색인이 기본 키 정의와 일치하는지를 판별하기 위해 점검이 수행됩니다(색인의 모든 INCLUDE 컬럼은 무시합니다.). 컬럼의 순서나 방향(ASC/DESC) 스펙에 관계없이 동일한 컬럼 세트를 식별할 경우, 색인 정의가 일치합니다. 일치하는 색인 정의가 있는 경우, 시스템에 요구되는 기본 색인임을 나타내도록 색인의 설명이 변경되고, 고유하지 않은 색인일 경우 (고유성을 확인한 후) 고유한 색인으로 변경됩니다. 테이블에 일치하는 색인이 두 개 이상 있을 경우, 기존의 고유 색인이 선택됩니다(선택은 임의적입니다.). 일치하는 색인이 없을 경우, CREATE TABLLE에 기술된 것과 같이 컬럼에 대해 고유성 색인이 자동으로 작성됩니다. 고유성 제한조건과 연관된 색인 이름에 대한 547 페이지의 『주』에서 자세한 내용을 참조하십시오.

하나의 테이블에서 단 하나의 기본 키만 정의할 수 있습니다.

ADD *참조 제한조건*

참조 제한조건을 정의합니다. 800 페이지의 『CREATE TABLE』의 *참조 제한조건*에서 자세한 내용을 참조하십시오.

ADD *점검 제한조건*

점검 제한조건을 정의합니다. 800 페이지의 『CREATE TABLE』의 *점검 제한조건*에서 참조하십시오.

ADD *파티션 키 정의*

파티션 키를 정의합니다. 테이블은 단일 파티션 노드그룹상의 테이블 공간에서 정의되어야 하며 파티션 키를 이미 가지고 있어서는 안 됩니다. 파티션 키가 이미 테이블에 대해 존재할 경우, 기존 키는 새로운 파티션 키를 추가하기 전에 삭제해야 합니다.

서브테이블인 테이블에 파티션 키를 추가할 수 없습니다(SQLSTATE 428DH).

PARTITIONING KEY(*컬럼 이름*...)

지정된 컬럼을 사용하여 파티션 키를 정의합니다. *각 컬럼 이름*은 테이블의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별되어서는 안 됩니다. 이름을 규정화할 수 없습니다. 컬럼의 데이터 유형이 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, 이 유형들 중 하나에 대한 구별 유형, 또는 구조화 유형일 경우 그 컬럼은 파티션 키의 부분으로 사용할 수 없습니다.

USING HASHING

데이터 분배를 위한 파티션 방법으로 해싱 함수의 사용을 지정합니다. 이것이 유일하게 지원되는 파티션 방법입니다.

ALTER *컬럼 교체*

컬럼의 특성을 변경합니다.

컬럼 이름

테이블에서 변경될 컬럼 이름입니다. *컬럼 이름*은 테이블의 기존 컬럼을 식별해야 합니다(SQLSTATE 42703). 이름을 규정화할 수 없습니다.

SET DATA TYPE VARCHAR(*정수***)**

기존 VARCHAR 컬럼의 길이를 증가합니다. VARCHAR 키워드에 대한 동의어로서 CHARACTER VARYING 또는 CHAR VARYING을 사용할 수 있습니다. *컬럼 이름*의 데이터 유형은 VARCHAR이어야 하고, 컬럼에 대한 현재 최대 길이는 *정수* 값보다 커서는 안 됩니다(SQLSTATE 42837). *정수*의 값 범위는 32672까지 가능합니다. 테이블은 입력된 테이블이어서는 안 됩니다(SQLSTATE 428DH).

컬럼을 변경해도 모든 컬럼의 총 바이트 수가 1258 페이지의 표34에서 지정된 최대 레코드 크기를 초과해서는 안 됩니다(SQLSTATE 54010). 849 페이지의 『주』에서 좀더 자세한 정보를 참조하십시오. 고유 제한조건이나 색인에서 컬럼이 사용될 경우, 새로운 길이는 255 바이트를 초과하면 안 되고 고유 제한조건이나 색인에 대한 저장 길이 합이 1024를 초과하면 안 됩니다(SQLSTATE 54008)(저장 길이에 대해서는 853 페이지의 『바이트 계수』에서 자세한 내용을 참조하십시오.).

SET EXPRESSION AS (*생성-표현식***)**

컬럼에 대한 표현식을 지정된 *생성 표현식*으로 변경합니다. SET EXPRESSION AS는 SET INTEGRITY문을 사용하여 테이블이 점검 보류 상태가 되도록 요구합니다. ALTER TABLE문 다음에, SET INTEGRITY문을 사용하여 새 표현식에 대해 해당 컬럼의 모든 값을 갱신하고 점검해야 합니다. 컬럼은 표현식을 근거로 이미 생성된 컬럼으로 정의되어 있어야 합니다(SQLSTATE 42837). 생성 표현식은 생성된 컬럼을

정의할 때 적용되는 것과 같은 규칙을 따라야 합니다. 생성 표현식의 결과 데이터 유형은 컬럼의 데이터 유형에 지정할 수 있어야 합니다(SQLSTATE 42821).

ADD SCOPE

아직 영역이 정의되어 있지 않은 기존의 참조 유형 컬럼에 영역을 추가합니다(SQLSTATE 428DK). 교체중인 테이블이 입력된 테이블인 경우, 그 컬럼은 수퍼 테이블로부터 물려받아서는 안 됩니다(SQLSTATE 428DJ). 예를 보려면 561 페이지의 『ALTER TYPE(구조화)』에서 참조하십시오.

입력된 테이블 이름

입력된 테이블의 이름. *컬럼 이름*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *입력된 테이블 이름* 유형입니다(SQLSTATE 428DM). 값이 *입력된 테이블 이름*에 있는 기존 행들을 실제로 참조하는지 확인하기 위해 *컬럼 이름*에 있는 기존 값들에 대해 점검을 하지 않습니다.

입력된 뷰 이름

입력된 뷰의 이름. *컬럼 이름*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *입력된 뷰 이름* 유형입니다(SQLSTATE 428DM). 값이 *입력된 뷰 이름*에 있는 기존 행들을 실제로 참조하는지 확인하기 위해 *컬럼 이름*에 있는 기존 값들에 대해 점검을 하지 않습니다.

DROP PRIMARY KEY

기본 키의 정의와 이 기본 키에 종속되는 모든 참조 제한조건을 삭제합니다. 테이블에는 기본 키가 있어야 합니다.

DROP FOREIGN KEY *제한조건 이름*

참조 제한조건의 *제한조건 이름*을 삭제합니다. *제한조건 이름*은 참조 제한조건을 식별해야 합니다. 참조 제한조건의 삭제에 대해서는 547 페이지의 『주』에서 정보를 참조하십시오.

DROP UNIQUE *제한조건 이름*

고유성 *제한조건 이름*의 정의와 이 고유성 제한조건에 종속되는 모든 참조 제한조건을 삭제합니다. *제한조건 이름*은 기존의 고유성(UNIQUE) 제한조건을 식별해야 합니다. 고유성 제한조건 삭제에 내포된 사항에 대해서는 547 페이지의 『주』에서 자세한 내용을 참조하십시오.

DROP CONSTRAINT *제한조건 이름*

제한조건의 *제한조건 이름*을 삭제합니다. *제한조건 이름*은 테이블에 정의된 기존의 점검 제한조건, 참조 제한조건, 기본 키 또는 고유성 제한조건을 식별해야 합니다. 제한조건의 삭제에 대해서는 547 페이지의 『주』에서 자세한 내용을 참조하십시오.

DROP CHECK *제한조건 이름*

점검 제한조건의 *제한조건 이름*을 삭제합니다. *제한조건 이름*은 테이블에 정의된 기존의 점검 제한조건을 식별해야 합니다.

DROP PARTITIONING KEY

파티션 키를 삭제합니다. 테이블에는 파티션 키가 있어야 하며, 이 테이블은 단일 파티션 노드 그룹에 정의된 테이블 공간이 있어야 합니다.

DATA CAPTURE

데이터 복제를 추가의 정보가 로그에 기록되는 지를 나타냅니다.

테이블이 유형화된 테이블이라면, 이 옵션이 지원되지 않습니다(루트 테이블의 경우에는 SQLSTATE 428DH, 기타 서브테이블의 경우에는 428DR).

NONE

추가의 정보가 기록되지 않음을 나타냅니다.

CHANGES

이 테이블에 대한 SQL 변경사항에 따라 추가의 정보가 로그에 기록됨을 나타냅니다. 이 옵션은 이 테이블이 전달되며 캡쳐 프로그램이 로그로부터 이 테이블에 대한 변경사항을 캡쳐하는 데 사용할 경우에 필요합니다.

테이블이 카탈로그 파티션(카탈로그 파티션 이외의 다른 파티션으로 된 복수의 파티션 노드 그룹 또는 노드 그룹) 이외의 다른 파티션에서 데이터를 허용하도록 정의된 경우는 이 옵션이 지원되지 않습니다(SQLSTATE 42997).

테이블의 스키마 이름(내재적 또는 암시적)이 18 바이트보다 길 경우, 이 옵션은 지원되지 않습니다(SQLSTATE 42997).

복사 사용에 대한 정보는 *관리 안내서*와 *복제 안내 및 참조서*에서 찾을 수 있습니다.

INCLUDE LONGVAR COLUMNS

데이터 복제 유틸리티가 LONG VARCHAR 또는 LONG VARGRAPHIC 컬럼에 이루어진 변경사항을 캡쳐할 수 있도록 합니다. 절은 테이블에 컬럼이 포함되도록 변경할 수 있으므로 LONG VARCHAR 또는 LONG VARGRAPHIC 컬럼이 없는 테이블에 대해 절을 지정할 수 있습니다.

ACTIVATE NOT LOGGED INITIALLY

현재 작업 단위에 대한 테이블의 NOT LOGGED INITIALLY 속성을 활성화합니다. 이 테이블은 원래 NOT LOGGED INITIALLY 속성으로 작성되었을 것입니다(SQLSTATE 429AA).

이 명령문에 의해 테이블이 교체된 후 동일한 작업 단위에서 INSERT, DELETE, UPDATE, CREATE INDEX, DROP INDEX 또는 ALTER TABLE에 의한 테이블 변경사항은 기록되지 않습니다. NOT LOGGED INITIALLY 속성이 활성화되는 ALTER문에 의한 시스템 카탈로그 변경사항은 기록되지 않습니다. 동일한 작업 단위 내에서 이후의 시스템 카탈로그 정보 변경사항은 기록됩니다.

현재 작업 단위가 완료되면 NOT LOGGED INITIALLY 속성은 비활성화되고, 이후의 작업 단위에서 테이블에서 수행되는 모든 연산은 기록됩니다.

데이터를 삽입하면서 카탈로그 테이블에서의 잠금을 방지하기 위해 이 피처를 사용하는 경우, ALTER TABLE문에서 이 절만 지정되어야 합니다. ALTER TABLE문에서 다른 절을 사용하면 카탈로그 잠금이 됩니다. ALTER TABLE 문에 대해 다른 절이 지정되지 않는 경우, 시스템 카탈로그 테이블에서 SHARE 잠금만 됩니다. 이렇게 하면 이 명령문이 실행되어 실행된 작업 단위가 종료될 때까지의 시간 동안 동시성 충돌이 일어날 가능성이 현저히 줄어들게 됩니다.

테이블이 유형화된 테이블이라면, 이 옵션이 유형화된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

NOT LOGGED INITIALLY 속성에 대한 자세한 내용은 800 페이지의 『CREATE TABLE』에 있는 이 속성에 대한 설명을 보십시오.

주: 작업 단위(UOW) 내에서 NOT LOGGED INITIALLY 속성을 활성화하여 테이블을 변경한 경우, 저장 지점까지의 구간 복원 요청은 작업 단위(UOW)까지의 구간 복원 요청으로 변환됩니다(SQLSTATE 40506). NOT LOGGED INITIALLY 속성이 사용중인 작업 단위 내에서의 연산 오류가 발생하면 전체 작업 단위가 구간 복원될 수 있습니다(SQLSTATE 40506). 또한, 구간 복원이 발생한 후, 초기에 로그되지 않은 속성이 활성화된 테이블은 **액세스 불가**로 표시되고 삭제만 가능합니다. 따라서, NOT LOGGED INITIALLY 속성이 사용중인 작업 단위 내에서 오류가 발생할 가능성은 최소로 해야 합니다.

WITH EMPTY TABLE

테이블에 현재 있는 모든 데이터가 제거됩니다. 일단 데이터가 제거되면 RESTORE 기능으로만 복구할 수 있습니다. 이 Alter문이 발행된 작업 단위가 구간 복원되는 경우, 테이블 데이터는 원래 상태로 되지 않습니다.

이 조치가 요청되면 영향받은 테이블에 정의된 DELETE 트리거는 발생하지 않습니다. 테이블에 있는 색인도 비게 됩니다.

PCTFREE *정수*

각 페이지에 대해 로드 또는 재조직 중에 빈 공간으로 남길 비율을 표시합니다. *정수* 값의 범위는 0에서 99까지입니다. 제한 없이 각 페이지의 첫번째 행이 추가됩니다. 행이 추가될 때 각 페이지에 적어도 *정수* 퍼센트의 빈 공간이 남게 됩니다. PCTFREE 값은 LOAD 및 REORGANIZE TABLE 유틸리티에 의해서만 고려됩니다. 테이블이 유형화된 테이블이라면, 이 옵션이 유형화된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

LOCKSIZE

테이블이 액세스될 때 사용되는 잠금 *크기*(granularity)를 표시합니다. 테이블 정의에서 이 옵션을 사용하면 일반적인 잠금 레벨 자동 업그레이드 발생이 방해되지 않습니다. 테이블이 유형화된 테이블이라면, 이 옵션이 유형화된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

ROW

행 잠금 사용을 표시합니다. 이는 테이블이 작성될 때 기본 잠금 *크기*입니다.

TABLE

테이블 잠금 사용을 표시합니다. 이는 테이블상에서 적절한 공유 또는 독점 잠금이 이루어지고 intent 잠금(intent none 제외)이 사용되지 않습니다. 이 값을 사용하면 설정되어야 할 잠금 수를 제한하여 조회 성능이 향상될 수 있습니다. 그러나, 완전한 테이블을 통해 모든 잠금이 유지되어 있으므로 동시성도 감소됩니다.

잠금에 대한 자세한 정보는 *관리 안내서*에 있습니다.

APPEND

데이터 페이지에서 가용 공간이 사용 가능한 곳에 데이터를 둘 것인지 테이블 데이터 끝부분에 추가할 것인지 표시합니다. 테이블이 유형화된 테이블이라면, 이 옵션이 유형화된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

ON

테이블 데이터를 추가할 것과, 페이지의 빈 공간에 대한 정보를 보유하지 않을 것을 표시합니다. 테이블에는 클러스터 색인이 없어야 합니다(SQLSTATE 428CA).

OFF

테이블 데이터가 사용 가능한 공간에 놓일 것을 나타냅니다. 이는 테이블이 작성될 때 기본값입니다.

사용 가능한 빈 공간에 대한 정보가 정확하지 않아 삽입시 성능이 저하될 수 있으므로 APPEND OFF를 설정한 후에는 테이블을 재구성해야 합니다.

VOLATILE

이는 최적화 알고리즘에게 테이블 *table-name*의 기본 행수(cardinality)가 런타임에 빈 것에서 아주 큰 것에 이르기까지 크게 다를 수 있음을 나타냅니다. 해당 색인이 색인 전용이거나(참조된 모든 컬럼이 색인에 있음) 해당 색인이 색인 스캔에서 술어를 적용할 수 있다면, 최적화 알고리즘은 통계에 관계없이 *table-name*을 액세스하기 위해 테이블 스캔보다는 색인 스캔을 사용할 것입니다. 테이블이 유형화된 테이블이라면, 이 옵션이 유형화된 테이블 계층의 루트 테이블에 대해서만 지원됩니다(SQLSTATE 428DR).

NOT VOLATILE

이는 최적화 알고리즘에게 *table-name*의 기본 행수(cardinality)가 휘발성이지 않음을 나타냅니다. 이 테이블에 대한 액세스 플랜은 계속 기존 통계 및 최적화 레벨에 기초할 것입니다.

CARDINALITY

휘발성이며 테이블 자체에 있지 않은 테이블에 있는 행 수임을 나타내기 위한 선택적 키워드.

규칙

- 테이블의 파티션 키 컬럼은 갱신할 수 없습니다(SQLSTATE 42997).
- 테이블에 정의된 고유 키 또는 기본 키 제한조건은 한 개일 경우, 파티션 키의 수퍼 세트여야 합니다(SQLSTATE 42997).
- 파티션 키의 널(NULL) 값 입력 가능 컬럼은 ON DELETE SET NULL을 사용하여 관계가 정의되면 외부 키 컬럼으로 포함시킬 수 없습니다(SQLSTATE 42997).
- 단일 ALTER TABLE문에 있는 하나의 ADD 또는 ALTER COLUMN절에서 컬럼은 참조만 될 수 있습니다(SQLSTATE 42711).
- 테이블에 종속된 요약 테이블이 있으면 테이블의 컬럼 길이는 변경될 수 없습니다(SQLSTATE 42997).
- 생성된 컬럼을 추가하기 전에, 테이블은 SET INTEGRITY문을 사용하여 점검 보류 상태로 설정해야 합니다(SQLSTATE 55019).

주

- 테이블을 요약 테이블로 변경하면 테이블이 점검 보류 상태에 놓이게 됩니다. 테이블이 REFRESH IMMEDIATE로 정의될 경우, 테이블은 INSERT, DELETE 또는 UPDATE 명령이 fullselect에 의해 참조되는 테이블에서 호출되기 전에 점검 보류 상태에서 벗어나야 합니다. 테이블은 IMMEDIATE CHECKED 옵션과 함께 REFRESH TABLE 또는 SET INTEGRITY를 사용하여 점검 보류 상태를 벗어나서, fullselect를 기초로 테이블의 데이터를 완전하게 새로 고칠 수 있습니다. 테이블의 데이터가 fullselect의 결과를 정확하게 반영할 경우,

SET INTEGRITY의 IMMEDIATE UNCHECKED 옵션을 사용하여 테이블 상태를 점검 보류 상태에서 벗어나게 할 수 있습니다.

- 테이블을 REFRESH IMMEDIATE 요약 테이블로 변경하면 fullselect에 의해 참조되는 테이블에서 INSERT, DELETE 또는 UPDATE가 사용되는 패키지가 유효하지 않게 됩니다.
- 테이블을 요약 테이블에서 일반 테이블로 변경하면(DEFINITION ONLY) 그 테이블에 종속되는 패키지가 유효하지 않게 됩니다.
- ADD 컬럼 절은 모든 다른 절보다 먼저 처리됩니다. 다른 절들은 지정된 순서대로 처리됩니다.
- ALTER TABLE을 통해 추가되는 임의의 컬럼은 기존 테이블의 뷰에 자동으로 추가되지 않습니다.
- 고유성 제한조건이나 기본 키 제한조건에 대해 색인이 자동으로 작성되는 경우, 데이터베이스 관리 프로그램은 테이블의 스키마 이름과 일치하는 스키마 이름과 더불어 지정된 제한조건 이름을 색인 이름으로 사용하려 합니다. 이것이 기존의 색인 이름과 일치하거나 제한조건에 대한 이름이 지정되지 않은 경우, SYSIBM 스키마에 색인이 작성되며 시스템 생성 이름은 ″SQL″과 이에 후속하는 시각 함수 생성에 의해 생성된 15자리 숫자 문자로 이루어집니다.
- 테이블 T상의 DELETE 조작에 수반될 수도 있는 테이블을 T에 *연속 삭제*된다고 말합니다. 그러므로, 이것이 T의 종속이거나 T 연쇄로부터 삭제하는 테이블의 종속이라면 테이블은 T에 연속 삭제됩니다.
- 레코드가 패키지 내의 명령문에 의해 직접적으로 또는 명령문의 하나를 대신해서 패키지에 의해 수행되는 제한조건 또는 트리거를 통해 간접적으로 T로 삽입(갱신/삭제)되는 경우, 패키지에서는 테이블 T에 대한 삽입(갱신/삭제) 사용을 갖고 있습니다. 유사하게, 컬럼이 패키지 내의 명령문에 의해 직접 갱신되거나 패키지 명령문의 하나를 대신하여 패키지에 의해 실행되는 제한조건 또는 트리거를 통해 간접적으로 갱신되는 경우, 패키지에서는 컬럼상의 갱신 사용을 갖고 있습니다.
- 기본 키, 고유 키 또는 외부 키에 대한 변경은 패키지, 색인 및 기타 외부 키에 다음과 같은 영향을 미칠 수 있습니다.
  - 기본 키나 고유 키가 추가되는 경우,

  - 패키지, 외부 키 또는 기본의 고유 키에는 영향을 미치지 않습니다.[61]
- 기본 키나 고유 키가 삭제되는 경우,
  - 제한조건에 대해 색인이 자동으로 작성된 경우, 색인이 삭제됩니다. 그 색인에 종속된 모든 패키지는 무효화됩니다.
  - 색인이 제한조건에 대해 고유한 것으로 변환된 경우 이는 다시 비고유 색인으로 되돌려지며, 더 이상 시스템 필수 색인이 아닙니다. 그 색인에 종속된 모든 패키지는 무효화됩니다.
  - 색인이 제한조건에 대해 사용된 기존의 고유 색인인 경우, 그 색인은 더 이상 시스템 필수 색인으로 설정되지 않습니다. 패키지에는 아무 영향을 주지 않습니다.
  - 모든 종속 외부 키가 삭제됩니다. 다음 항목에 지정된 대로, 각 종속 외부 키에 대해 추가 조치가 취해집니다.
- 외부 키가 추가 또는 삭제되는 경우
  - 오브젝트 테이블에 대해 삽입 사용을 갖고 있는 모든 패키지가 유효하지 않게 됩니다.
  - 외부 키 내에서 최소한 한 컬럼에 갱신 사용을 갖고있는 모든 패키지는 유효하지 않게 됩니다.
  - 상위 테이블에서 삭제 사용을 갖고 있는 모든 패키지가 유효하지 않게 됩니다.
  - 상위 키내에서 적어도 한 컬럼에 대해 갱신 사용된 적이 있는 모든 패키지는 무효화됩니다.

- 테이블에 컬럼을 추가하면 변경된 테이블에 삽입이 사용된 적이 있는 모든 패키지가 무효화됩니다. 추가된 컬럼이 테이블에서 사용자가 정의한 첫번째 구조화 유형 컬럼일 경우, 변경된 테이블에서 DELETE를 사용하는 패키지도 유효하지 않게 됩니다.
- 이미 존재하고 점검 보류 상태(1160 페이지의 『SET INTEGRITY』 참조)에 있지 않는 테이블에 점검 제한조건 또는 참조 제한조건을 추가하면, 테이블에 있는 기존 행이 제한조건에 대해 즉시 평가됩니다. 확인이 실패하면, 오류

61. 기본 키 또는 고유 키가 이전 버전에서 생성된 기존의 고유 색인을 사용하고 전수된 고유성을 지원하도록 변환되지 않은 경우, 색인은 변환되며 관련 테이블에 갱신이 사용된 패키지는 무효화됩니다.

(SQLSTATE 23512)가 발생합니다. 테이블이 점검 보류 상태에 있으면, 점검 제한조건 또는 참조 제한조건이 즉시 제한조건을 시행하지는 않습니다. 그 대신, 점검 보류 작업에서 사용되는 해당 제한조건 유형 플래그가 갱신됩니다. 제한조건을 시행하려면, SET INTEGRITY문을 발행해야 합니다.

- 점검 제한조건을 추가하거나 삭제하면, 오브젝트 테이블상에 삽입 사용을 갖고 있거나 점검 제한조건이 적용되는 컬럼 중 최소한 컬럼에 갱신 사용을 갖고 있는 모든 패키지는 유효하지 않게 됩니다.
- 파티션 키를 추가하면 파티션 키의 컬럼 중 최소한 한 컬럼에서 갱신을 사용하는 모든 패키지가 유효하지 않게 됩니다.
- 기본 키의 첫번째 컬럼으로 정의된 파티션 키는 기본 키를 삭제하거나 다른 기본 키를 추가하는 작업의 영향을 받지 않습니다.
- 컬럼의 길이를 증가시키면, 변경된 컬럼이 있는 테이블을 참조(참조 제한조건 및 트리거를 통해 직간접적으로)하는 모든 패키지를 무효화합니다.
- 컬럼의 길이를 증가시키면 테이블에 종속된 뷰(입력된 뷰는 제외)를 재생성합니다. 뷰를 재생성하는 데 오류가 발생하면, 오류가 리턴됩니다(SQLSTATE 56098). 테이블에 종속된 모든 입력된 뷰는 작동하지 않는 것으로 표시됩니다.
- 컬럼을 변경하여 길이를 증가시키면 트리거가 포함되는 명령문이 준비되거나 바인드될 때 트리거 처리 시 오류가 발생할 수 있습니다(SQLSTATE 54010). 전환 변수 및 전환 테이블 컬럼 길이 합계에 기초하여 행 길이가 너무 길 때 이렇게 될 수 있습니다. 그러한 트리거가 삭제된 경우 그 뒤로 이를 작성하려고 하면 오류가 발생합니다(SQLSTATE 54040).
- 각각 4000과 2000보다 길게 되도록 변경된 VARCHAR 및 VARGRAPHIC 컬럼은 SYSFUN 스키마에서 함수의 입력 매개변수로 사용하지 말아야 합니다(SQLSTATE 22001).
- 테이블의 LOCKSIZE를 변경하면 교체된 테이블에 종속성이 있는 모든 패키지가 무효화됩니다. 잠금에 대한 자세한 정보는 *관리 안내서*에 있습니다.
- FILE LINK CONTROL 속성을 가진 DATALINK 컬럼이 속성에 추가되고 있을 때에는 ACTIVATE NOT LOGGED INITIALLY절이 사용될 수 없습니다(SQLSTATE 42613).
- VOLATILE이나 NOT VOLATILE CARDINALITY를 변경하면 변경된 테이블에 종속성을 가지는 모든 패키지가 뮤효화될 것입니다.

- 복제 고객은 VARCHAR 컬럼의 길이를 증가시킬 때 주의를 기울여야 합니다. 응용프로그램 테이블과 연관된 데이터 변경 테이블이 이미 DB2 rowsize 한계에 있거나 여기에 가까이 있을 지도 모릅니다. 두 테이블 모두에 대해 변경이 완료될 수 있으려면 응용프로그램 테이블에 앞서 데이터 변경 테이블이 변경되거나, 동일한 작업 단위(UOW) 내에서 두 개가 변경되어야 합니다. rowsize 한계에 있거나 이에 가까이 있거나 또는 기존 컬럼의 길이를 증가시키는 기능이 부족한 플랫폼에 상주하는 사본에 대해서는 주의를 기울여야 합니다.

  Capture 프로그램이 증가된 VARCHAR 컬럼 길이로 로그 레코드를 처리하기 전에 데이터 변경 테이블이 변경되지 않으면, Capture 프로그램이 실패할 수도 있습니다. VARCHAR 컬럼을 포함하고 있는 사본이 사본 수행을 유지보수하는 복사 작업 내역에 앞서 변경되지 않으면, 복사 작업 내역이 실패할 수도 있습니다.

예

예 1: 한 문자 길이의 새 컬럼 RATING을 DEPARTMENT 테이블에 추가합니다.

```
ALTER TABLE DEPARTMENT
 ADD RATING CHAR(1)
```

예 2: SITE_NOTES라고 하는 새 컬럼을 PROJECT 테이블에 추가합니다. 최대 길이가 1000자인 가변 길이 컬럼으로 SITE_NOTES를 작성합니다. 컬럼의 값은 연관되는 문자 집합을 갖고 있지 않으므로, 변환될 수 없습니다.

```
ALTER TABLE PROJECT
 ADD SITE_NOTES  VARCHAR(1000) FOR BIT DATA
```

예 3: EQUIPMENT라고 하는 테이블이 다음 컬럼으로 정의되어 존재한다고 가정합시다.

컬럼 이름	데이터 유형
EQUIP_NO	INT
EQUIP_DESC	VARCHAR(50)
LOCATION	VARCHAR(50)
EQUIP_OWNER	CHAR(3)

소유자(EQUIP_OWNER)가 DEPARTMENT 테이블에 있는 부서 번호(DEPTNO)가 되도록 참조 제한조건을 EQUIPMENT 테이블에 추가합니다.

DEPTNO는 DEPARTMENT 테이블의 기본 키입니다. 부서가 DEPARTMENT 테이블에서 제거된 경우, 그 부서에서 소유하는 모든 장비에 대한 소유자 (EQUIP_OWNER) 값은 지정되지 않게 되어야 합니다(또는, 널(NULL)로 설정). 제한조건에 DEPTQUIP 이름을 제공합니다.

```
ALTER TABLE EQUIPMENT
 ADD CONSTRAINT DEPTQUIP
  FOREIGN KEY (EQUIP_OWNER)
    REFERENCES DEPARTMENT
     ON DELETE SET NULL
```

또한, 추가 컬럼에서는 이 장치 레코드와 연관된 양을 기록하는 것을 허용해야 합니다. 달리 지정하지 않으면, EQUIP_QTY 컬럼의 값은 1이고 널(NULL)이 아니어야 합니다.

```
ALTER TABLE EQUIPMENT
ADD COLUMN EQUIP_QTY
SMALLINT NOT NULL DEFAULT 1
```

예 4: EMPLOYEE 테이블을 교체합니다. 각 사원에 대해 급여와 수당의 합계가 $30,000을 초과하도록 정의된 점검 제한조건 REVENUE를 추가합니다.

```
ALTER TABLE EMPLOYEE
ADD CONSTRAINT REVENUE
CHECK (SALARY + COMM > 30000)
```

예 5: EMPLOYEE 테이블을 교체합니다. 이전에 정의된 제한조건 REVENUE를 삭제합니다.

```
ALTER TABLE EMPLOYEE
DROP CONSTRAINT REVENUE
```

예 6: SQL 변경사항을 기본 형식으로 로그하도록 테이블을 변경합니다.

```
ALTER TABLE SALARY1
   DATA CAPTURE NONE
```

예 7: SQL 변경사항을 확장 형식으로 로그하도록 테이블을 변경합니다.

```
ALTER TABLE SALARY2
   DATA CAPTURE CHANGES
```

예 8: 기본값을 갖는 4개의 새 컬럼을 추가하도록 EMPLOYEE 테이블을 변경합니다.

```
ALTER TABLE EMPLOYEE
ADD COLUMN HEIGHT   MEASURE   DEFAULT MEASURE(1)
ADD COLUMN BIRTHDAY BIRTHDATE DEFAULT DATE('01-01-1850')
ADD COLUMN FLAGS    BLOB(1M)  DEFAULT BLOB(X'01')
ADD COLUMN PHOTO    PICTURE   DEFAULT BLOB(X'00')
```

기본값은 기본값 지정시 다양한 함수 이름을 사용합니다. MEASURE는 INTEGER에 기초한 구별 유형이므로, MEASURE 함수가 사용됩니다. HEIGHT 컬럼 기본값은 MEASURE의 소스 유형이 BLOB 또는 날짜시간 데이터 유형이 아니므로, 함수 없이 지정되었을 수 있습니다. BIRTHDATE가 DATE에 기초한 구별 유형이므로, DATE 함수가 사용됩니다 (여기에는 BIRTHDATE가 사용될 수 없음). FLAGS 및 PHOTO 컬럼의 경우, 기본값은 PHOTO가 구별 유형이더라도 BLOB 함수를 사용하여 지정됩니다. BIRTHDAY, FLAGS 및 PHOTO 컬럼의 기본값을 지정하려면, 유형이 BLOB나 BLOB 또는 날짜 시간 데이터 유형에 기초한 구별 유형이므로 함수가 사용되어야 합니다.

예 9: 다음과 같은 컬럼이 정의된 CUSTOMERS라고 하는 테이블이 있다고 가정합시다.

```
컬럼 이름         데이터 유형
BRANCH_NO           SMALLINT
CUSTOMER_NO         DECIMAL(7)
CUSTOMER_NAME       VARCHAR(50)
```

이 테이블에서, 기본 키는 BRANCH_NO 및 CUSTOMER_NO 컬럼으로 구성됩니다. 테이블을 파티션하고자 하므로, 테이블에 대해 파티션 키를 작성해야 합니다. 테이블은 단일 노드로 된 노드 그룹의 테이블 공간에 정의되어야 합니다. 기본 키는 파티션 컬럼의 수퍼 세트여야 합니다. 기본 키의 컬럼들 중 최소한 하나가 파티션 키로 사용되어야 합니다. BRANCH_NO를 파티션 키로 만들려고 할 경우, 다음 명령문에서 이를 지정하면 됩니다.

```
ALTER TABLE CUSTOMERS
      ADD PARTITIONING KEY (BRANCH_NO)
```

ALTER TABLESPACE

ALTER TABLESPACE문은 다음과 같은 방법으로 기존의 테이블 공간을 수정하는 데 사용됩니다.

- 컨테이너를 DMS 테이블 공간(즉, MANAGED BY DATABASE 옵션으로 작성된 테이블 공간)에 추가합니다.
- DMS 테이블 공간(즉, MANAGED BY DATABASE 옵션으로 작성된)에서 컨테이너 *크기*를 증가시킵니다.
- 현재 컨테이너를 가지고 있지 않은 파티션(또는 노드)상의 SMS 테이블 공간에 컨테이너를 추가합니다.
- 테이블 공간에 대해 PREFETCHSIZE 설정값을 수정합니다.
- 테이블 공간에서 테이블에 대해 사용된 BUFFERPOOL을 수정합니다.
- 테이블 공간에 대해 OVERHEAD 설정값을 수정합니다.
- 테이블 공간에 대해 TRANSFERRATE 설정값을 수정합니다.

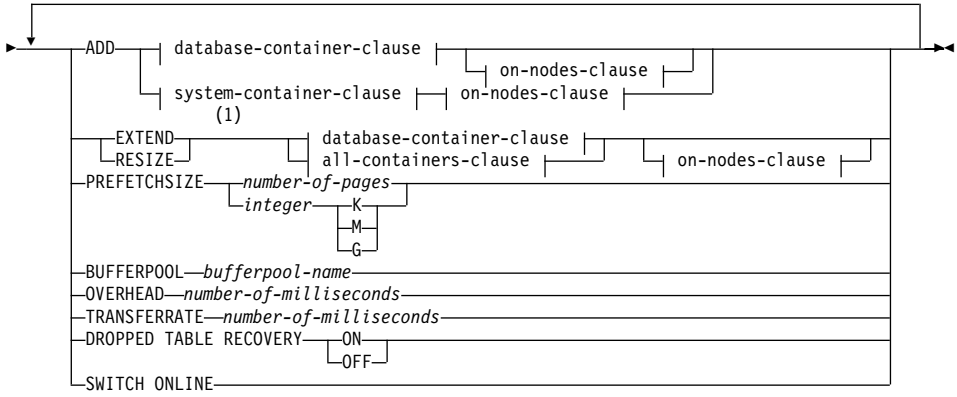
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

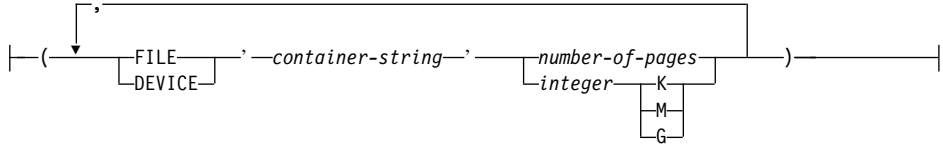
권한 부여

명령문의 권한 부여 ID는 SYSCTRL 또는 SYSADM 권한이 있어야 합니다.

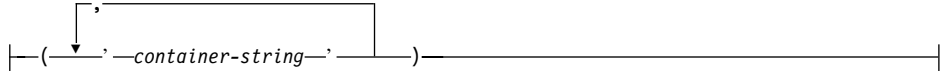
구문



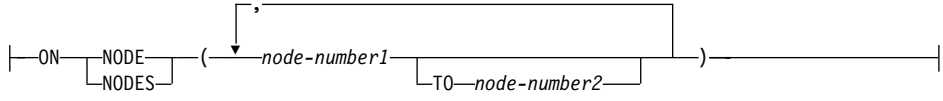
database-container-clause:



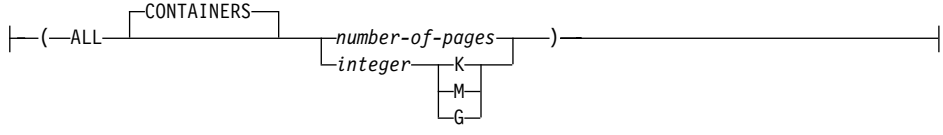
system-container-clause:



on-nodes-clause:



all-containers-clause:



주:

1 같은 명령문에 ADD, EXTEND 및 RESIZE 절을 지정할 수 없습니다.

설명

테이블 공간 이름

테이블 공간 이름. 이 이름은 한 부분의 이름입니다. 이 이름은 긴 SQL 식별자입니다(일반 식별자 또는 분리 식별자입니다.).

ADD

ADD는 새로운 컨테이너가 테이블 공간에 추가됨을 지정합니다.

EXTEND

EXTEND는 *기존 컨테이너*가 *크기*에서 증가됨을 지정합니다. 지정된 *크기*는 기존 컨테이너가 증가되는 단위 *크기*입니다. *모든 컨테이너 절*을 지정할 경우, 테이블 공간에 있는 모든 컨테이너가 이 *크기*만큼씩 증가됩니다.

RESIZE

RESIZE는 *기존 컨테이너*의 *크기*가 변경됨을 지정합니다(컨테이너 *크기*만 증가될 수 있습니다). 지정된 *크기*는 컨테이너의 새 *크기*입니다. *모든 컨테이너 절*을 지정할 경우, 테이블 공간에 있는 모든 컨테이너가 이 *크기*로 변경됩니다.

데이터베이스 컨테이너 절

하나 이상의 컨테이너를 DMS 테이블 공간에 추가합니다. 테이블 공간은 응용프로그램 서버에 이미 존재하는 DMS 테이블 공간을 식별해야 합니다. 페이지 865에 있는 *컨테이너 절*의 설명을 참조하십시오.

시스템 컨테이너 절

하나 이상의 컨테이너를 지정된 파티션이나 노드상의 SMS 테이블 공간에 추가합니다. 테이블 공간은 응용프로그램 서버(AS)에 이미 존재하는 SNS 테이블 공간을 식별해야 합니다. 테이블 공간에 대해 지정된 파티션에 컨테이너가 있으면 안 됩니다. (SQLSTATE 42921). 페이지 864에 있는 *시스템 컨테이너*의 설명을 참조하십시오.

on-노드절

추가된 컨테이너에 대해 파티션이나 파티션들을 지정합니다. 페이지 866에 있는 *on-노드 절*의 설명을 참조하십시오.

모든 컨테이너 절

DMS 테이블 공간에 있는 모든 컨테이너를 확장하거나 *크기*를 조정합니다. 테이블 공간은 응용프로그램 서버에 이미 존재하는 DMS 테이블 공간을 식별해야 합니다.

PREFETCHSIZE *페이지 수*

데이터 프리페칭이 수행되는 동안 테이블 공간으로부터 읽어 들일 PAGESIZE 페이지의 수를 지정합니다. 프리페치 *크기* 값은 정수로도 지정할 수 있으며 숫자 뒤에 K(킬로바이트의 경우), M(메가바이트의 경우) 또는 G(기가바이트의 경우)가 붙습니다. 이런 방법으로 지정될 경우, 바이트 수를 페이지 *크기*로 나눈 최저값은 프리페치 *크기*에 대한 페이지 수를 결정하는 데 사용됩니다. 프리페칭은 조회에서 참조되기 전에 조회에서 필요한 데이터를 읽어서, I/O 수행 시 조회가 지연되지 않도록 합니다.

BUFFERPOOL *버퍼 풀 이름*

이 테이블 공간의 테이블에 사용된 버퍼 풀의 이름. 버퍼 풀이 현재 데이터베이스에 있어야 합니다(SQLSTATE 42704). 테이블 공간의 노드 그룹은 버퍼 풀에 대해 정의되어야 합니다(SQLSTATE 42735).

OVERHEAD *밀리세컨드*

I/O 제어기 오버헤드 및 디스크 탐색 및 대기 시간을 밀리초 단위로 지정하는 숫자 문자형 상수(정수, 소수 또는 부동 소수점). 이 숫자는 모든 컨테이너에 대해 같지 않을 경우, 테이블 공간에 속하는 모든 컨테이너에 대한 평균이어야 합니다. 이 값은 조회 최적화시 I/O의 비용을 계산할 때 사용됩니다.

TRANSFERRATE *밀리세컨드*

메모리로 한 페이지(4K 또는 8K)를 읽어들이는 시간(밀리초)을 지정하는 숫자 리터럴(정수, 십진수 또는 부동 소수). 이 숫자는 모든 컨테이너에 대해 같지 않을 경우, 테이블 공간에 속하는 모든 컨테이너에 대한 평균이어야 합니다. 이 값은 조회 최적화시 I/O의 비용을 계산할 때 사용됩니다.

DROPPED TABLE RECOVERY

지정된 테이블 공간에서 삭제된 테이블은 ROLLFORWARD 명령의 RECOVER DROPPED TABLE ON 옵션을 복구될 수 있습니다.

SWITCH ONLINE

OFFLINE 상태의 테이블 공간은 컨테이너가 액세스 가능하게 될 경우에 온라인 상태가 됩니다. 컨테이너가 액세스 가능해지지 않으면 오류가 리턴됩니다(SQLSTATE 57048).

주

- PREFETCHSIZE, OVERHEAD 및 TRANSFERRATE 매개변수에 대한 최적의 값을 선택하기 위한 지침과 재조정 관련 정보는 *관리 안내서*에서 제공합니다.
- 새로운 컨테이너가 추가되었고 트랜잭션이 확약되면, 테이블 공간의 내용은 컨테이너에서 자동으로 평균화됩니다. 테이블 공간에 대한 액세스는 다시 평균화하는 동안 제한되지 않습니다.
- 테이블 공간이 OFFLINE 상태에 있으며 컨테이너가 액세스 가능해 졌으면, 사용자가 모든 응용프로그램을 연결해제하고 다시 데이터베이스에 연결하여 테이블 공간을 OFFLINE 상태로부터 가져올 수 있습니다. 대안으로, SWITCH ONLINE 옵션이 데이터베이스의 나머지가 여전히 시동되어 사용되고 있는 동안에도 테이블 공간을 (OFFLINE으로부터) 가져올 수 있습니다.
- 둘 이상의 컨테이너를 테이블 공간에 추가할 경우, 재조정 비용이 한 번만 발생하도록 컨테이너를 동일한 명령문으로 추가하는 것이 좋습니다. 단일 트랜잭션 내에서 별도의 ALTER TABLESPACE문으로 동일한 테이블 공간에 컨테이너를 추가하면 오류가 발생합니다(SQLSTATE 55041).
- 테이블 공간에서 컨테이너 *크기*를 변경할 수 없으며 새 컨테이너가 같은 ALTER TABLESPACE문에 추가될 수 없습니다(SQLSTATE 429BC). 여러 컨테이너의 *크기*를 변경할 때, 하나의 명령문에 EXTEND 절과 RESIZE 절을 동시에 사용할 수 없습니다(SQLSTATE 429BC).
- RESIZE는 컨테이너 *크기*를 감소시킬 때 사용할 수 없습니다. 컨테이너에 대해 더 작은 *크기*를 지정하려고 하면 오류가 발생합니다(SQLSTATE 560B0).

- 존재하지 않는 컨테이너를 확장하거나 *크기*를 조정하려고 하면 오류가 발생합니다(SQLSTATE 428B2).
- 컨테이너를 확장하거나 *크기*를 조정할 때, 컨테이너 유형은 컨테이너가 작성될 때 사용된 유형과 일치해야 합니다(SQLSTATE 428B2).
- 컨테이너가 *호가장*되거나 *크기*가 조정된 후 트랜잭션이 확약되고 나면, 테이블 공간의 내용은 컨테이너들 사이에서 자동으로 균형 있게 나눠집니다. 테이블 공간에 대한 액세스는 다시 평균화하는 동안 제한되지 않습니다.
- 여러 컨테이너를 테이블 공간에서 확장할 경우, 재조정 비용이 한 번만 발생하도록 컨테이너가 동일한 명령문에서 변경되는 것이 좋습니다. 이는 여러 컨테이너 *크기*를 재조정할 경우에도 마찬가지입니다. 단일 트랜잭션 내에서 별도의 ALTER TABLESPACE문으로 동일한 테이블 공간에서 컨테이너 *크기*를 변경하려고 하면 오류가 발생합니다(SQLSTATE 55041).
- 파티션된 데이터베이스에서, 두 개 이상의 파티션이 물리적으로 동일한 노드에 상주할 경우, 그러한 파티션에 대해 동일한 장치나 특정 경로를 지정할 수 없습니다(SQLSTATE 42730). 이러한 환경에서는 각 파티션에 대해 고유한 *컨테이너 문자열*을 지정하거나 상대 경로 이름을 사용하십시오.
- 버퍼 풀 정의가 트랜잭션이 가능하며, 확약시 테이블 공간 정의에 대한 변경이 카탈로그 테이블에 반영된다고 하더라도, 다음 번에 데이터베이스가 시작될 때까지는 새로운 정의를 가진 버퍼 풀을 사용할 수 없습니다. ALTER TABLESPACE문이 실행될 때 사용중인 버퍼 풀은 이 중간에도 계속 사용됩니다.

예

예 1: 장치를 PAYROLL 테이블 공간에 추가합니다.

```
ALTER TABLESPACE PAYROLL
   ADD (DEVICE '/dev/rhdisk9' 10000)
```

예 2: ACCOUNTING 테이블 공간에 대한 프리페치 *크기*와 I/O 오버헤드를 변경합니다.

```
ALTER TABLESPACE ACCOUNTING
   PREFETCHSIZE 64
   OVERHEAD 19.3
```

예 3: 테이블 공간 TS1을 작성한 후, 모든 컨테이너의 페이지 수가 2000이 되도록 *크기*를 조정합니다(이 *크기* 조정을 수행할 세 가지의 다른 ALTER TABLESPACES가 제공됩니다).

```
CREATE TABLESPACE TS1
   MANAGED BY DATABASE
   USING (FILE '/conts/cont0' 1000,
          DEVICE '/dev/rcont1' 500,
          FILE 'cont2' 700)
ALTER TABLESPACE TS1
   RESIZE (FILE '/conts/cont0' 2000,
           DEVICE '/dev/rcont1' 2000,
           FILE 'cont2' 2000)
```

또는

```
ALTER TABLESPACE TS1
    RESIZE (ALL 2000)
```

또는

```
ALTER TABLESPACE TS1
    EXTEND (FILE '/conts/cont0' 1000,
            DEVICE '/dev/rcont1' 1500,
            FILE 'cont2' 1300)
```

예 4: DATA_TS 테이블 공간에서 모든 컨테이너를 1000 페이지씩 확장합니다.

```
ALTER TABLESPACE DATA_TS
   EXTEND (ALL 1000)
```

예 5: INDEX_TS 테이블 공간에서 모든 컨테이너의 *크기*를 100 메가바이트(MB)로 조정합니다.

```
ALTER TABLESPACE INDEX_TS
   RESIZE (ALL 100 M)
```

ALTER TYPE(구조화)

ALTER TYPE문은 사용자 정의 구조 유형의 속성 또는 메소드 스펙을 추가하거나 삭제하기 위해 사용됩니다.

호출

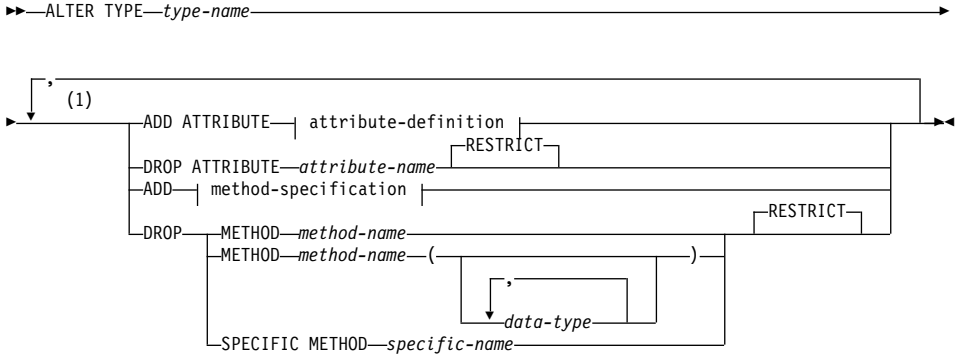
이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비할 수 있는 실행 가능한 명령문입니다. 그러나 DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문을 동적으로 준비할 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권에 다음 중 최소한 하나는 포함되어야 합니다.

- SYSADM 또는 DBADM 권한
- 유형 스키마에 대한 ALTERIN 특권
- SYSCAT.DATATYPES의 DEFINER 컬럼에 기록된 유형의 정의자

구문



주:

1 속성과 메소드 둘다가 추가되거나 삭제될 경우, 모든 메소드 스펙 이전에 모든 속성 스펙이 발생해야 합니다.

설명

유형 이름

변경될 구조화 유형을 식별합니다. 카탈로그에 정의된 기존의 유형(SQLSTATE 42704)이어야 하고, 유형은 구조화 유형(SQLSTATE 428DP)이어야 합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

ADD ATTRIBUTE

기존의 구조화 유형의 최종 속성 다음에 속성을 추가합니다.

속성 정의

*속성 정의*의 세부 설명에 대해서는 893 페이지의 『CREATE TYPE(구조화)』에서 자세한 내용을 참조하십시오.

속성 이름

속성에 대한 이름을 지정합니다. 이름은 해당되는 구조화 유형의 다른 속성(계승된 속성을 포함하여)이나, 해당되는 구조화 유형의 부속 유형에 대한 다른 속성과 같으면 안 됩니다(SQLSTATE 42711).

술어에서 키워드로 사용되는 여러 이름들은 시스템에서 사용하도록 예약되어 있으므로, *속성 이름*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH와 같은 이름들이 있고, 비교 연산자도 포함됩니다.

데이터 유형 1

속성의 데이터 유형을 지정합니다. 이것은 LONG VARCHAR, LONG VARGRAPHIC, 또는 LONG VARCHAR이나 LONG VARGRAPHIC을 기초로 하는 구별 유형 이외에, CREATE TABLE 아래에 나열되는 데이터 유형 중 하나입니다(SQLSTATE 42601). 데이터 유형은 기존의 데이터 유형을 식별해야 합니다(SQLSTATE 42704). 스키마 이름 없이 *데이터 유형*을 지정할 경우, SQL 경로의 스키마를 검색하여 유형이 해석됩니다. 다양한 데이터 유형에 대한 설

명은 800 페이지의 『CREATE TABLE』에 있습니다. 속성 데이터 유형이 참조 유형일 경우, 참조의 목표 유형은 존재하는 구조화 유형이어야 합니다(SQLSTATE 42704).

속성 유형이 DATALINK로 정의된 구조화 유형은 입력된 테이블이나 입력 뷰에 대한 데이터 유형으로만 효율적으로 사용할 수 있습니다(SQLSTATE 01641).

런타임에서 유형의 인스턴스에 직접 또는 간접적으로 같은 유형의 다른 인스턴스나 해당되는 부속 유형 중 하나가 포함되도록 하는 유형 정의를 금지하기 위해, 해당되는 속성 유형 중 하나가 직접 또는 간접적으로 자체를 사용하는 것과 같이 유형이 정의되지 않도록 하는 제한사항이 있습니다(SQLSTATE 428EP). 102 페이지의 『구조화 유형』에서 자세한 설명을 참조하십시오.

lob 옵션

LOB 유형과 연관된 옵션(또는 LOB 유형에 기초한 구별 유형)을 지정합니다. lob 옵션의 세부 설명에 대해서는 800 페이지의 『CREATE TABLE』에서 자세한 내용을 참조하십시오.

데이터 링크 옵션

DATALINK 유형과 연관된 옵션(또는 DATALINK 유형에 기초한 구별 유형)을 지정합니다. 데이터링크 옵션의 세부 설명에 대해서는 800 페이지의 『CREATE TABLE』에서 자세한 내용을 참조하십시오.

DATALINK 유형이나 DATALINK에서 전래되는 구별 유형에 대해 어떤 옵션도 지정하지 않을 경우, 기본값은 LINKTYPE URL 및 NO LINK CONTROL 옵션입니다.

DROP ATTRIBUTE

기존의 구조화 유형 속성을 삭제합니다.

속성 이름

속성의 이름. 속성은 유형의 속성으로서 존재해야 합니다(SQLSTATE 42703).

RESTRICT

기존의 테이블, 뷰, 컬럼, 컬럼 유형 내에 중첩된 속성, 또는 색인 확장의 유형으로 *유형 이름*을 사용할 경우 어떤 속성도 삭제될 수 없게 하는 규칙이 시행됩니다.

ADD 메소드 스펙

유형 이름에 의해 식별되는 유형에 메소드 스펙을 추가합니다. 메소드는 별도의 CREATE METHOD문을 사용하여 메소드에 내용을 제공할 때까지 사용할 수 없습니다. 메소드 스펙에 대해서는 893 페이지의 『CREATE TYPE(구조화)』에서 자세한 내용을 참조하십시오.

DROP METHOD

삭제될 메소드의 인스턴스를 식별합니다. 지정된 메소드는 기존 메소드 내용을 가지고 있지 않아야 합니다(SQLSTATE 428ER). DROP METHOD문을 사용하여, ALTER TYPE DROP METHOD를 사용하기 이전에 메소드 내용을 삭제하도록 하십시오.

지정된 메소드는 카탈로그에 설명된 메소드여야 합니다(SQLSTATE 42704). CREATE TYPE문에 의해 내재적으로 생성되는 메소드(예: mutators 및 observers)는 삭제할 수 없습니다(SQLSTATE 42917).

삭제될 메소드 스펙을 식별할 수 있는 몇 가지의 방법이 있습니다.

METHOD *메소드 이름*

특별한 메소드를 식별하고, 이름이 *메소드 이름*이고 주제 유형이 *유형 이름*인 메소드 인스턴스가 정확히 하나 있는 경우에만 유효합니다. 그러므로 식별되는 메소드에는 임의 개수의 매개변수가 있을 수 있습니다. 유형 *유형 이름*에 대해 이 이름의 메소드가 전혀 존재하지 않을 경우, 오류가 발생합니다(SQLSTATE 42704). 명명된 데이터 유형에 대해 이름이 *메소드 이름*인 하나 이상의 메소드가 있을 경우, 오류가 발생합니다(SQLSTATE 42854).

METHOD *메소드 이름(데이터 유형,...)*

삭제할 메소드를 고유하게 식별하는 메소드 시그니처를 제공합니다. 메소드 선택 알고리즘은 사용되지 않습니다.

메소드 이름

특정 유형에 대해 삭제될 메소드의 이름. 이름은 고유하지 않은 식별자여야 합니다.

(데이터 유형,...)

메소드가 정의될 때 메소드 스펙의 해당 위치에서 지정한 데이터 유형과 일치해야 합니다. 데이터 유형의 수와 데이터 유형의 논리적 병합은 삭제될 특정 메소드 인스턴스를 식별하는 데 사용됩니다.

매개변수화된 데이터 유형에 대한 정밀도 또는 스케일, 길이를 지정하는 것은 필요하지 않습니다. 대신, 데이터 유형 일치를 찾을 때 이들 속성이 무시됨을 표시하기 위해 빈 괄호를 쓸 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용될 수 없습니다(SQLSTATE 42601).

그러나, 길이, 정밀도 또는 스케일이 코드화된 경우, 그 값은 CREATE TYPE문에 지정된 것과 정확히 일치해야 합니다.

0<n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치(Matching)는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 시그니처에 해당되는 메소드가 명명된 데이터 유형에 대해 존재하지 않으면, 오류가 발생합니다(SQLSTATE 42883).

SPECIFIC METHOD *특정 이름*

메소드가 정의될 때 제공되거나 기본값으로 설정된 특정 이름을 사용하여, 삭제될 특정 메소드를 식별합니다. *특정 이름*이 규정되지 않은 이름일 경우, 메소드는 *유형 이름*에 대해 지정된 데이터 유형의 스키마에서 내재적으로 규정됩니다. 특정 이름은 유형 *유형 이름*에 대해 메소드를 식별해야 하며, 그렇지 않으면 오류가 발생합니다(SQLSTATE 42704).

RESTRICT

지정된 메소드가 기존 메소드 내용을 갖는 데 있어서 제한됨을 나타냅니다. DROP METHOD문을 사용하여, ALTER TYPE DROP METHOD를 사용하기 이전에 메소드 내용을 삭제하도록 하십시오.

규칙

- 속성을 추가하거나 삭제하는 것은 다음 중 한 경우에 해당될 때 유형 *유형 이름*에 대해 허용되지 않습니다(SQLSTATE 55043).
  - 유형이나 해당되는 부속 유형 중 하나가 기존 테이블이나 뷰의 유형입니다. 또는,
  - 해당 유형이 직접 또는 간접으로 *유형 이름*을 사용하는 테이블의 컬럼이 존재합니다. *직접 사용*과 *간접 사용*이라는 용어는 102 페이지의 『구조화 유형』에 정의되어 있습니다.
  - 유형이나 해당되는 부속 유형 중 하나가 색인 확장에서 사용됩니다.
- 유형은 유형이나 해당되는 부속 유형 중 어느 하나의 총 속성 수가 4082를 초과하도록 속성을 추가하여 변경할 수는 없습니다(SQLSTATE 54050).
- ADD ATTRIBUTE 옵션:
  - ADD ATTRIBUTE는 새로운 속성에 대한 observer 및 mutator 메소드를 생성합니다. 이 메소드들은 893 페이지의 『CREATE TYPE(구조화)』에 설명된 것처럼 구조화 유형이 작성될 때 생성된 것과 유사합니다. 이 메소드들이 기존 메소드나 함수와 상충되거나 이를 대체할 경우, ALTER TYPE문이 실패합니다(SQLSTATE 42745).
  - 유형(또는 해당되는 부속 유형)에 대한 INLINE LENGTH가 값이 292 이하인 사용자에 의해 명시적으로 지정되었고, 추가된 속성으로 인해 지정된 인라인 길이가 변경된 유형에 대한 constructor 함수의 결과 크기(속성마다 32 바이트 + 10 바이트)보다 작게 될 경우, 오류가 발생합니다(SQLSTATE 42611).
- DROP ATTRIBUTE 옵션:
  - 기존의 수퍼 유형에서 계승된 속성은 삭제할 수 없습니다(SQLSTATE 428DJ).
  - DROP ATTRIBUTE는 삭제된 속성의 mutator 및 observer 메소드를 삭제한 후 삭제된 메소드에 대한 종속성을 확인합니다.

주

- 속성 추가 또는 삭제로 유형이 변경될 경우, 이 유형이나 이 유형의 부속 유형을 매개변수나 결과로 사용하는 함수나 메소드의 영향을 받는 모든 패키지가 유효하지 않게 됩니다.
- 속성이 구조화 유형에 추가되거나 구조화 유형에서 삭제될 경우:
  - 유형의 INLINE LENGTH가, 유형이 작성될 때 시스템에서 계산된 경우, INLINE LENGTH 값은 변경된 유형과, 변경하려는 해당되는 모든 부속 유형에 대해 자동으로 수정됩니다. INLINE LENGTH 값은 또한 시스템에서 INLINE LENGTH가 계산되었고 유형에 INLINE LENGTH가 변경된 유형의 속성이 포함되는 모든 구조화 유형에 대해 자동으로(반복적으로) 수정됩니다.
  - 속성 추가 또는 삭제에 의해 영향을 받는 유형의 INLINE LENGTH를 사용자가 명시적으로 정의하였으면, 그 특정 유형에 대한 INLINE LENGTH는 변경되지 않습니다. 명시적으로 지정된 인라인 길이에 대해 특히 주의해야 합니다. 나중에 유형에 속성이 추가될 가능성이 있으면, 컬럼 정의에서 유형이나 이에 해당되는 부속 유형의 사용을 위해, 인라인 길이는 인스턴스화된 오브젝트의 길이에서 증가 가능성을 고려할만큼 충분히 커야 합니다.
  - 새로운 속성이 응용프로그램에 보여지게 되면, 기존 변환 함수는 데이터 유형의 새 구조에 일치되도록 수정되어야 합니다.

예

예 1: ALTER TYPE문을 사용하여 상호 참조 유형 및 테이블 순환을 허용할 수 있습니다. EMPLOYEE 및 DEPARTMENT 상호 참조 테이블을 고려하십시오.

다음 연속은 유형 및 테이블이 작성되도록 합니다.

```
CREATE TYPE DEPT ...
CREATE TYPE EMP ... (유형 REF(DEPT)의 DEPTREF 속성 포함)
ALTER TYPE DEPT ADD ATTRIBUTE MANAGER REF(EMP)
CREATE TABLE DEPARTMENT OF DEPT ...
CREATE TABLE EMPLOYEE OF EMP (DEPTREF WITH OPTIONS SCOPE DEPARTMENT)
ALTER TABLE DEPARTMENT ALTER COLUMN MANAGER ADD SCOPE EMPLOYEE
```

다음 연속은 이들 테이블 및 유형이 삭제되도록 합니다.

```
DROP TABLE EMPLOYEE   (DEPARTMENT의 MANAGER 컬럼의 영역 지정이 취소되었습니다)
DROP TABLE DEPARTMENT
ALTER TYPE DEPT DROP ATTRIBUTE MANAGER
DROP TYPE EMP
DROP TYPE DEPT
```

예 2: ALTER TYPE문을 사용하여 부속 유형을 참조하는 속성을 가진 유형을 작성할 수 있습니다.

```
CREATE TYPE EMP ...
CREATE TYPE MGR UNDER EMP ...
ALTER TYPE EMP ADD ATTRIBUTE MANAGER REF(MGR)
```

예 3: ALTER TYPE문을 사용하여 속성을 추가할 수 있습니다. 다음 명령문은 SPECIAL 속성을 EMP 유형에 추가합니다. 원래의 CREATE TYPE문에 인라인 길이가 지정되지 않았으므로, DB2는 13을 더하여 인라인 길이를 다시 계산합니다(새 속성에 대한 10 바이트 + 속성 길이 + 2 LOB 이외 속성에 대한 2 바이트).

```
ALTER TYPE EMP ...
   ADD ATTRIBUTE SPECIAL CHAR(1)
```

예 4 ALTER TYPE문을 사용하여 유형과 연관되는 메소드를 추가할 수 있습니다. 다음 명령문은 BONUS라고 하는 메소드를 추가합니다.

```
ALTER TYPE EMP ...
   ADD METHOD BONUS (RATE DOUBLE)
      RETURNS INTEGER
      LANGUAGE SQL
      CONTAINS SQL
      NO EXTERNAL ACTION
      DETERMINISTIC
```

BONUS 메소드는 CREATE METHOD문을 발행하여 메소드 내용을 작성할 때까지 사용할 수 없습니다. 유형 EMP에 SALARY라고 하는 속성이 포함되어 있다고 가정할 경우, 다음은 메소드 내용 정의의 예입니다.

```
CREATE METHOD BONUS(RATE DOUBLE) FOR EMP
   RETURN CAST(SELF.SALARY * RATE AS INTEGER)
```

이 명령문에 대해서는 758 페이지의 『CREATE METHOD』에서 자세한 내용을 참조하십시오.

ALTER USER MAPPING

지정된 연합 서버 권한 부여 ID를 위한 데이터 소스에서 사용되는 권한 부여 ID나 암호를 변경하는 데 ALTER USER MAPPING문이 사용됩니다.

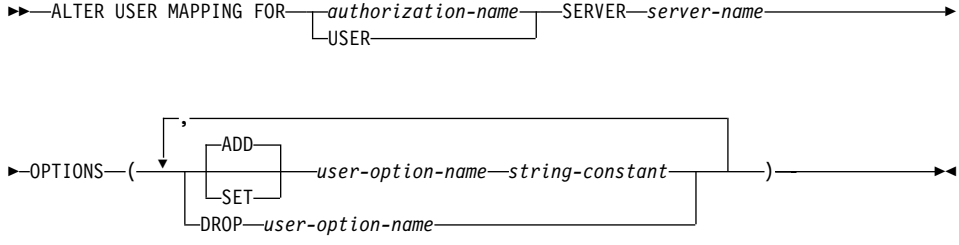
호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID가 데이터 소스에 맵핑되는 권한 부여 이름과 다르면, 명령문의 권한 부여 ID가 SYSADM 또는 DBADM 권한을 포함해야 합니다. 그렇지 않으면, 권한 부여 ID 및 권한 부여 이름이 일치하는 경우 어떠한 특권도 필요하지 않습니다.

구문



설명

권한 부여 이름

사용자나 응용프로그램이 연합 데이터베이스에 연결하는 권한 부여 이름을 지정합니다.

USER

특수 레지스터 USER에 있는 값. USER가 지정될 때에는, ALTER USER

MAPPING문의 권한 부여 ID가 REMOTE_AUTHID 사용자 옵션에서 지정되는 데이터 소스 권한 부여 ID로 맵핑됩니다.

SERVER *서버 이름*

*authorization-name*으로 표시되거나 USER에 의해 참조되는 지역 권한 부여 ID로 맵핑하는 원격 권한 부여 ID하에서 액세스 가능한 데이터 소스를 식별합니다.

OPTIONS

변경되는 맵핑에 대해 어느 사용자 옵션이 작동가능, 재설정 또는 삭제될 지를 나타냅니다. *사용자 옵션 이름* 및 그 설정값에 대한 설명은 1412 페이지의 『사용자 옵션』에서 참조하십시오.

ADD

사용자 옵션을 작동가능하게 합니다.

SET

사용자 옵션의 설정값을 변경합니다.

사용자 옵션 이름

작동가능화되거나 재설정될 사용자 옵션에 이름을 부여합니다.

문자열 상수

*user-option-name*에 대한 설정값을 문자열 상수로서 지정합니다.

DROP *사용자 옵션 이름*

사용자 옵션을 삭제합니다.

주

- 사용자 옵션은 동일한 ALTER USER문에서 두번 이상 지정될 수 없습니다 (SQLSTATE 42853). 사용자 옵션이 작동가능화, 재설정 또는 삭제될 때, 사용중인 다른 서버 옵션들에는 영향이 미치지 않습니다.
- UOW가 이미 맵핑에 포함될 데이터 소스에서 테이블이나 뷰에 대한 별명 (nickname)을 참조하는 SELECT문을 포함하는 경우에는 주어진 작업 단위 (UOW)에서 사용자 맵핑이 변경될 수 없습니다.

예

예 1: Jim은 지역 데이터베이스를 사용하여 ORACLE1이라는 Oracle 데이터 소스에 연결합니다. 그는 권한 부여 ID KLEEWEIN으로 지역 데이터베이스를 액세스합니다. KLEEWEIN은 Jim이 ORACLE1을 액세스하는 권한 부여 ID인 CORONA로 맵핑됩니다. Jim은 새로운 ID인 JIMK로 ORACLE1을 액세스하기 시작할 것입니다. KLEEWEIN이 이제 JIMK로 맵핑되어야 합니다.

```
ALTER USER MAPPING FOR KLEEWEIN
   SERVER ORACLE1
   OPTIONS ( SET REMOTE_AUTHID 'JIMK' )
```

예 2: Mary는 연합 데이터베이스를 사용하여 DORADO라는 OS/390용 DB2 Universal Database 데이터 소스에 연결합니다. 그녀는 한 권한 부여 ID를 사용하여 DB2에 액세스하고 또 다른 권한 부여 ID를 사용하여 DORADO에 액세스합니다. 그리고 이들 두 ID간에 맵핑을 작성했습니다. 이들 ID 둘다에 동일한 암호를 사용하고 있었으나, DORADO에 대한 ID로 별도의 암호인 ZNYQ를 사용하기로 결정합니다. 따라서, 그녀는 그녀의 연합 데이터베이스 암호를 ZNYQ로 맵핑해야 합니다.

```
ALTER USER MAPPING FOR MARY
   SERVER DORADO
   OPTIONS ( ADD REMOTE_PASSWORD 'ZNYQ' )
```

ALTER VIEW

ALTER VIEW문은 참조 유형 컬럼을 교체하여 영역을 추가함으로써 기존 뷰를 수정합니다.

호출

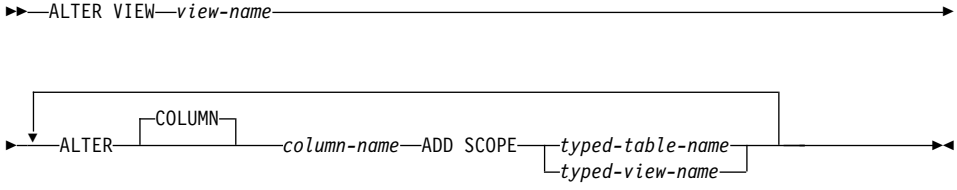
이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 뷰의 스키마에 대한 ALTERIN 특권
- 교체할 뷰의 정의자(definer)
- 교체할 뷰에 대한 CONTROL 특권.

구문



설명

뷰 이름

변경할 뷰를 식별합니다. 카탈로그에 기술된 뷰여야 합니다.

ALTER COLUMN *컬럼 이름*

뷰에서 교체될 컬럼의 이름입니다. *컬럼 이름*은 뷰의 기존 컬럼을 식별해야 합니다(SQLSTATE 42703). 이름을 규정화할 수 없습니다.

ADD SCOPE

아직 영역이 정의되어 있지 않은 기존의 참조 유형 컬럼에 영역을 추가합니다(SQLSTATE 428DK). 컬럼은 수퍼 뷰로부터 물려받아서는 안됩니다(SQLSTATE 428DJ).

입력된 테이블 이름

입력된 테이블의 이름. *컬럼 이름*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *입력된 테이블 이름* 유형입니다(SQLSTATE 428DM). 값이 *입력된 테이블 이름*에 있는 기존 행들을 실제로 참조하는지 확인하기 위해 *컬럼 이름*에 있는 기존 값들에 대해 점검을 하지 않습니다.

입력된 뷰 이름

입력된 뷰의 이름. *컬럼 이름*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *입력된 뷰 이름* 유형입니다(SQLSTATE 428DM). 값이 *입력된 뷰 이름*에 있는 기존 행들을 실제로 참조하는지 확인하기 위해 *컬럼 이름*에 있는 기존 값들에 대해 점검을 하지 않습니다.

BEGIN DECLARE SECTION

BEGIN DECLARE SECTION문은 호스트 변수 선언 섹션의 시작을 표시합니다.

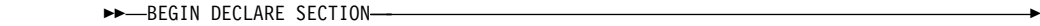
호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 실행되지 않습니다. REXX에 지정되어서는 안 됩니다.

권한 부여

필요 사항 없음.

구문



설명

BEGIN DECLARE SECTION문은 변수 선언이 호스트 언어의 규칙에 따라 나타날 수 있는 응용프로그램에서 코딩할 수 있습니다. 이것은 호스트 변수 선언 섹션의 시작을 표시하는 데 사용됩니다. 호스트 변수 섹션은 END DECLARE SECTION문으로 끝납니다(1014 페이지의 『END DECLARE SECTION』 참조).

규칙

- BEGIN DECLARE SECTION문 및 END DECLARE SECTION문은 쌍을 이루어야 하나, 중첩될 수는 없습니다.
- SQL문은 선언 절에 포함될 수 없습니다.
- SQL문에서 참조되는 변수는 REXX를 제외한 모든 호스트 언어에서 선언되어야 합니다. 또한, 그 섹션은 변수에 대한 첫번째 참조 전에 나타나야 합니다. 일반적으로, LOB 위치 지정자와 파일 참조 변수를 제외한 호스트 변수는 REXX에서는 선언할 수 없습니다. 이 경우, 이러한 변수는 BEGIN DECLARE SECTION 내에서 선언되지 않습니다.
- 선언 섹션 외부에서 선언된 변수는 선언 섹션 내에서 선언된 변수 이름과 같아서는 안 됩니다.

- LOB 데이터 유형은 해당되는 데이터 유형을 갖고 SQL TYPE IS 키워드가 앞에 오는 길이를 가져야 합니다.

예

예 1: 호스트 변수 hv_smint (smallint), hv_vchar24 (varchar(24)), hv_double (double), hv_blob_50k (blob(51200)), hv_struct(blob(10240)으로 구조화 유형 "struct_type"의)를 C 프로그램으로 정의합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
 short                          hv_smint;
 struct {
    short hv_vchar24_len;
    char  hv_vchar24_value[24];
   }                              hv_vchar24;
 double                         hv_double;
 SQL TYPE IS BLOB(50K) hv_blob_50k;
 SQL TYPE IS struct_type AS BLOB(10k) hv_struct;
EXEC SQL  END DECLARE SECTION;
```

예 2: COBOL 프로그램으로 호스트 변수 HV-SMINT(smallint), HV-VCHAR24(varchar(24)), HV-DEC72(dec(7,2)) 및 HV-BLOB-50k(blob (51200))를 정의합니다.

```
WORKING-STORAGE SECTION.
      EXEC SQL BEGIN DECLARE SECTION  END-EXEC.
01  HV-SMINT              PIC S9(4)       COMP-4.
01  HV-VCHAR24.
    49 HV-VCHAR24-LENGTH  PIC S9(4)       COMP-4.
    49 HV-VCHAR24-VALUE   PIC X(24).
01  HV-DEC72              PIC S9(5)V9(2)  COMP-3.
01 HV-BLOB-50K            USAGE SQL TYPE IS BLOB(50K).
      EXEC SQL END DECLARE SECTION  END-EXEC.
```

예 3: Fortran 프로그램으로 호스트 변수 HVSMINT(smallint), HVVCHAR24(char(24)), HVDOUBLE(double) 및 HVBLOB50k(blob(51200)) 를 정의합니다.

```
EXEC SQL  BEGIN DECLARE SECTION
  INTEGER*2      HVSMINT
  CHARACTER*24   HVVCHAR24
  REAL*8         HVDOUBLE
  SQL TYPE IS BLOB(50K)  HVBLOB50K
EXEC SQL  END DECLARE SECTION
```

주: Fortran에서, 예상되는 값이 254자를 초과하면, CLOB 호스트 변수를 사용해야 합니다.

예 4: REXX 프로그램으로 호스트 변수 HVSMINT(smallint), HVBLOB50K(blob(51200)) 및 HVCLOBLOC(CLOB 위치 지정자)을 정의합니다.

```
DECLARE :HVCLOBLOC LANGUAGE TYPE CLOB LOCATOR
call sqlexec 'FETCH c1 INTO :HVSMINT, :HVBLOB50K'
```

변수 HVSMINT 및 HVBLOB50K는 FETCH문에서 이것을 사용하여 내재적으로 정의되었음에 유의하십시오.

CALL

데이터베이스 위치에서 저장 프로시듀어를 호출합니다. 예를 들면, 저장 프로시듀어는 데이터베이스의 위치에서 실행되며, 데이터를 클라이언트 응용프로그램으로 리턴시킵니다.

SQL CALL문을 사용하는 프로그램은 두 부분, 즉 클라이언트 및 서버에서 수행되도록 설계되어 있습니다. 데이터베이스에서의 서버 프로시듀어는 클라이언트 응용프로그램과 같은 트랜잭션 내에서 수행됩니다. 클라이언트 응용프로그램과 저장 프로시듀어가 같은 파티션에 있으면, 저장 프로시듀어는 지역적으로 실행됩니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다. 그러나, USING DESCRIPTOR절의 사용과 함께, 호스트 변수와 CALL을 통해 지정될 수 있는 프로시듀어 이름은 프로시듀어 이름과 매개변수 목록을 런타임에 제공하도록 하여 동적으로 준비되는 명령문과 같은 효과를 얻게 됩니다.

권한 부여

권한 규칙은 프로시듀어가 저장되는 서버에 따라 다릅니다.

DB2 Universal Database:

명령문 **런타임** CALL문의 권한 부여 ID에서 갖고 있는 특권에는 최소한 다음 중 하나를 포함해야 합니다.

- 저장 프로시듀어와 연관되는 패키지에 대한 EXECUTE 특권
- 저장 프로시듀어와 연관되는 패키지에 대한 CONTROL 특권
- SYSADM 또는 DBADM 권한

OS/390용 DB2 Universal Database:

바인드시 CALL문의 권한 부여 ID에서 갖고 있는 특권에는 최소한 다음 중 하나를 포함해야 합니다.

- 저장 프로시듀어와 연관되는 패키지에 대한 EXECUTE 특권
- 저장 프로시듀어와 연관되는 패키지의 소유권
- 패키지의 콜렉션(collection)에 대한 PACKADM 권한

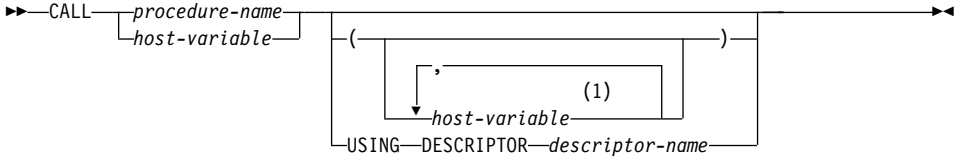
- SYSADM 권한

AS/400용 DB2

바인드시 CALL문의 권한 부여 ID에서 갖고 있는 특권에는 최소한 다음 중 하나를 포함해야 합니다.

- 저장 프로시듀어가 REXX로 작성되는 경우:
  - 프로시듀어와 연관되는 원시 파일에서의 시스템 권한 *OBJOPR 및 *READ
  - 원시 파일을 포함하는 라이브러리에 대한 시스템 권한 *EXECUTE와 CL 명령에 대한 시스템 권한 *USE
- 저장 프로시듀어가 REXX로 작성되지 않는 경우:
  - 프로시듀어와 연관되는 프로그램과 해당 프로그램을 포함하는 라이브러리 둘다에 대한 시스템 권한 *EXECUTE
- 관리 권한

구문



주:

1 OS/390용 DB2 Universal Database 및 AS/400용 DB2 Universal Database 서버에 위치하고 OS/390용 DB2 Universal Database 또는 AS/400용 DB2 Universal Database 클라이언트에 의해 호출되는 저장 프로시듀어는 프로시듀어 인수에 대해 추가 소스(예: 상수 값)를 지원합니다. 그러나, 저장 프로시듀어가 DB2 Universal Database에 위치하거나 프로시듀어가 DB2 Universal Database 클라이언트로부터 호출된 경우, 호스트 변수를 통해 모든 인수를 제공해야 합니다.

설명

프로시듀어 이름 또는 *호스트 변수*

호출할 프로시듀어를 식별합니다. 프로시듀어 이름은 직접 또는 호스트 변수 내에서 지정될 수 있습니다. 식별된 프로시듀어는 현재 서버에 존재해야 합니다 (SQLSTATE 42724).

*프로시듀어 이름*이 지정된 경우 254바이트보다 크지 않은 일반 식별자여야 합니다. 이 이름은 일반 식별자만 될 수 있으므로, 빈칸(blank)이나 특수 문자는 포함할 수 없으며 값은 대문자로 변환됩니다. 그러므로, 소문자, 빈칸 또는 특수 문자를 사용해야 하는 경우, 이름은 *호스트 변수*를 통해 지정해야 합니다.

*호스트 변수*를 지정하면, 254바이트보다 크지 않는 길이 속성을 갖고 있고 표시기 변수를 포함하지 않는 문자열 변수여야 합니다. 값은 대문자로 변환되지 **않는다**는 점에 유의하십시오. *프로시듀어 이름*은 왼쪽 정렬되어야 합니다.

프로시듀어 이름은 몇 가지 양식 중 하나를 취할 수 있습니다. 지원되는 양식은 프로시듀어가 저장되는 서버에 따라 다릅니다.

DB2 Universal Database:

프로시듀어 이름

실행될 프로시듀어의 이름(확장자없이). 호출되는 프로시듀어는 다음과 같이 결정됩니다.

1. *프로시듀어 이름*은 저장 프로시듀어 라이브러리의 이름과 그 라이브러리 내의 함수 이름으로 사용됩니다. 예를 들어, *프로시듀어 이름*이 `proclib`이면, DB2 서버는 `proclib`로 명명된 저장 프로시듀어 라이브러리를 로드하고 그 라이브러리 내에서 함수 루틴 `proclib()`를 실행합니다.

   UNIX 기반 시스템에서는, DB2 서버가 기본 디렉토리 `sqllib/function`에서 저장 프로시듀어 라이브러리를 찾습니다. 분리되지 않은 저장 프로시듀어는 `sqllib/function/unfenced` 디렉토리에 있습니다.

OS/2에서는, 저장 프로시듀어의 위치는 CONFIG.SYS 파일에서 LIBPATH 변수로 지정됩니다. 분리되지 않은 저장 프로시듀어는 SQLlib\dll\unfenced 디렉토리에 있습니다.

2. 라이브러리나 함수를 찾을 수 없는 경우, 일치하는 프로시듀어가 있는지 (SYSCAT.PROCEDURES에서) 정의 프로시듀어를 검색하는 데 *프로시듀어 이름*이 사용됩니다. 일치하는 프로시듀어는 다음 단계를 통해 결정됩니다.

   a. 카탈로그(SYSCAT.PROCEDURES)에서 PROCNAME이 지정된 *프로시듀어 이름*과 일치하고 PROCSCHEMA가 SQL 경로의 *스키마 이름*(CURRENT PATH 특수 레지스터)인 프로시듀어를 찾으십시오. *스키마 이름*이 명시적으로 지정되는 경우, SQL 경로는 무시되고 지정된 *스키마 이름*을 가진 프로시듀어만 찾습니다.

   b. 그런 다음, CALL문에 지정된 인수의 수와 매개변수의 수가 같지 않은 프로시듀어를 제거하십시오.

   c. SQL 경로의 맨 앞에 있는 나머지 프로시듀어를 선택하십시오.

   d. 단계 2를 수행한 후 남아 있는 프로시듀어가 없을 경우, 오류가 리턴됩니다(SQLSTATE 42884).

   일단 프로시듀어가 선택되면, DB2는 외부 이름을 통해 정의된 프로시듀어를 호출합니다.

프로시듀어 라이브러리!함수 이름

느낌표(!)는 라이브러리 이름과 저장 프로시듀어의 함수 이름 사이의 분리 문자로 사용됩니다. 예를 들어, proclib!func를 지정했으면, proclib가 메모리로 적

재되고 그 라이브러리로부터의 함수 func가 실행됩니다. 이로써, 여러 함수가 동일한 저장 프로시듀어 라이브러리에 들어 있을 수 있습니다.

저장 프로시듀어 라이브러리는 디렉토리 내에 있거나 *프로시듀어 이름*에서 설명된 것처럼 LIBATH 변수 내에서 지정될 수 있습니다.

절대 경로!함수 이름

*절대 경로*는 저장 프로시듀어 라이브러리에 대한 완전한 경로를 지정합니다.

예를 들어, UNIX 중심 시스템에서 /u/terry/proclib!func가 지정되었으면, 저장 프로시듀어 라이브러리 proclib는 디렉토리 /u/terry에서 확보되어, 그 라이브러리의 함수 func가 실행됩니다.

OS/2에서, d:\terry\proclib!func를 지정하였으면, 데이터베이스 관리 프로그램은 d:\terry\proclib 디렉토리에서 func.dll 파일을 로드합니다.

이러한 모든 경우에서, 내재적 또는 명시적 전체 경로를 포함하는 프로시듀어 이름의 총 길이는 254바이트보다 크지 말아야 합니다.

OS/390용 DB2 Universal Database (V4.1 이상) 서버:

내재적이거나 명시적인 세 개의 부분 이름. 그 부분들은 다음과 같습니다.

높은 순서: 프로시듀어가 저장되는 서버의 위치 이름.

중간: SYSPROC

중간: SYSIBM.SYSPROCEDURES 카탈로그 테이블의 PROCEDURE 컬럼에 있는 어떤 값.

OS/400용 DB2(V3.1 이후 버전) 서버:

외부 프로그램 이름은 *프로시듀어 이름*과 동일한 것으로 간주됩니다.

이식성에 대해, *프로시듀어 이름*은 8바이트의 단일 토큰으로 지정됩니다.

(호스트 변수,...)

호스트 변수의 각 스펙은 CALL의 매개변수입니다. CALL의 n번째 매개변수는 서버의 저장 프로시듀어의 n번째 매개변수에 해당됩니다.

각 호스트 변수는 클라이언트와 서버사이에서 양방향으로 데이터를 교환하는 데 사용되도록 가정됩니다. 클라이언트와 서버 사이에 불필요한 데이터가 송신되지 않도록, 클라이언트 응용프로그램에서는 각 매개변수에 표시기 변수를 제공해야 하고 매개변수가 저장 프로시듀어로 데이터를 전송하는 데 사용되지 않으면 -1로 설정해야 합니다. 저장 프로시듀어에서는 클라이언트 응용프로그램으로 데이터를 리턴하는 데 사용하지 않는 모든 매개변수에 대해서는 표시기 변수를 -128로 설정해야 합니다.

서버가 DB2 Universal Database일 경우, 매개변수는 클라이언트와 서버 프로그램에서 일치하는 데이터 유형을 가지고 있어야 합니다. [62]

USING DESCRIPTOR *설명자 이름*

호스트 변수의 유효한 설명이 포함되어 있는 SQLDA를 식별합니다. n번째 SQLVAR 요소는 서버의 저장 프로시듀어의 n번째 매개변수에 해당됩니다.

CALL문이 처리되기 전에, 응용프로그램은 SQLDA에 다음 필드를 설정해야 합니다.

- SQLDA에 제공되는 SQLVAR 어커런스 수를 나타내기 위한 SQLN.
- SQLDA에 대해 할당되는 저장영역의 바이트 수를 나타내기 위한 SQLDABC.
- 명령문 처리시 SQLDA에서 사용되는 변수의 수를 나타내기 위한 SQLDA.
- 변수의 속성을 나타내기 위한 SQLVAR 어커런스 수. 각 기본 SQLVAR 요소의 다음 필드를 초기설정해야 합니다.
  - SQLTYPE
  - SQLLEN
  - SQLDATA

62. OS/390용 DB2 Universal Database 및 AS/400용 DB2 Universal Database 서버는 저장 프로시듀어를 호출할 때 호환되는 데이터 유형간의 변환을 지원합니다. 예를 들어, 클라이언트 프로그램에서 INTEGER 데이터 유형을 사용하고 저장 프로시듀어에서는 FLOAT를 기대할 경우, 서버는 프로시듀어를 호출하기 전에 INTEGER 값을 FLOAT로 변환합니다.

– SQLIND

전달된 각 두 번째 SQLVAR 요소의 다음 필드를 초기설정해야 합니다.

– LEN.SQLLONGLEN

– SQLDATALEN

– SQLDATATYPE_NAME

각 SQLDA는 클라이언트와 서버 사이에서 양방향으로 교환되는 데이터용으로 사용되는 것으로 간주됩니다. 클라이언트와 서버 사이에 불필요한 데이터가 송신되지 않도록 하려면, 매개변수가 저장 프로시듀어로 데이터를 전송하는데 사용되지 않을 때, 클라이언트 프로그램에서는 SQLIND 필드를 -1로 설정해야 합니다. 저장 프로시듀어에서는 클라이언트 응용프로그램으로 데이터를 리턴하는 데 사용하지 않는 모든 매개변수에 대해서는 SQLIND 필드를 -128로 설정해야 합니다.

주

- *대형 오브젝트(LOB) 데이터 유형 사용*:

  클라이언트와 서버 응용프로그램이 SQLDA로부터 LOB 데이터를 지정해야 할 경우, SQLVAR 항목 수의 두 배를 할당해야 합니다.

  LOB 데이터 유형은 DB2 버전 2에서부터 저장 프로시듀어에서 지원됩니다. 그 아래 레벨의 모든 클라이언트나 서버에서는 LOB 데이터 유형이 지원되지 않습니다.

- *SQL 프로시듀어에서 RETURN_STATUS 검색*:

  SQL 프로시듀어가 상태 값과 함께 RETURN문을 성공적으로 발행할 경우, 이 값은 SQLCA의 첫번째 SQLERRD 필드에서 리턴됩니다. CALL문이 SQL 프로시듀어에서 발행되면, GET DIAGNOSTICS문을 사용하여 RETURN_STATUS 값을 검색하십시오. SQLSTATE가 오류를 나타낼 경우 값은 -1입니다.

- *저장 프로시듀어에서 결과 세트 리턴*:

  클라이언트 응용프로그램이 CLI를 이용하여 작성된 경우, 결과 집합은 직접 클라이언트 응용프로그램으로 리턴될 수 있습니다. 저장 프로시듀어는 그 결과 집

합상에서 커서를 선언하고 결과 집합상의 커서를 열고 프로시듀어를 나갈때 열린 커서를 나가서 결과 집합이 리턴됨을 나타냅니다.

CLI를 통해 호출된 프로시듀어의 종료시에는 다음을 수행합니다.

- 열려 있는 각 커서에 대해서는, 결과 집합이 응용프로그램으로 리턴됩니다.
- 둘 이상의 커서가 열려 있으면, 결과 집합은 커서가 열린 순서대로 리턴됩니다.
- 읽혀지지 않은 행만 다시 전달됩니다. 예를 들어, 커서의 결과 집합에 500개의 행이 있고 이 행중에서 150개의 행이 저장 프로시듀어에서 읽었으면, 저장 프로시듀어가 종료되는 시점에서 151부터 500까지의 행이 저장 프로시듀어로 리턴됩니다.

*응용프로그램 개발 안내서*와 *CLI Guide and Reference*에서 추가 정보를 참조하십시오.

- ***CALL문과 DARI API 사이의 상호작용성***:

  일반적으로, CALL문은 기존 DARI 프로시듀어와는 작동되지 않을 것입니다. *응용프로그램 개발 안내서*에서 자세한 내용을 참조하십시오.

- ***특수 레지스터 조절***:

  호출자의 특수 레지스터에 대한 설정은 호출 시 저장 프로시듀어에 의해 계승되고 호출자에게 리턴 시 복원됩니다. 특수 레지스터는 저장 프로시듀어 내에서 변경될 수 있지만, 이러한 변경사항이 호출자에게 영향을 주지는 않습니다. 이는 프로시듀어의 특수 레지스터에 대해 수행된 변경사항이 호출자에 대한 설정이 되는 레거시 저장 프로시듀어(매개변수 스타일 DB2DARI로 정의되거나 기본 라이브러리에서 발견되는 저장 프로시듀어)의 경우에는 적용되지 않습니다.

예

예 1:

C로, ACHIEVE 라이브러리에 있는 TEAMWINS라고 하는 프로시듀어를 호출하여 HV_ARGUMENT 호스트 변수에 저장되는 매개변수에 전달합니다.

```
strcpy(HV_PROCNAME, "ACHIEVE!TEAMWINS");
CALL :HV_PROCNAME (:HV_ARGUMENT);
```

예 2:

C에서, INOUT_SQLDA라고 하는 SQLDA를 사용하여 :SALARY_PROC라고 하는 프로시듀어를 호출합니다.

```
  struct sqlda *INOUT_SQLDA;
/* Setup code for SQLDA variables goes here */
 CALL :SALARY_PROC
 USING DESCRIPTOR :*INOUT_SQLDA;
```

예 3:

Java 저장 프로시듀어는 다음 명령문을 사용하여 데이터베이스에서 작성됩니다.

```
CREATE PROCEDURE PARTS_ON_HAND (IN  PARTNUM  INTEGER,
                                OUT COST     DECIMAL(7,2),
                                OUT QUANTITY INTEGER)
                EXTERNAL NAME 'parts!onhand'
                LANGUAGE JAVA PARAMETER STYLE DB2GENERAL;
```

Java 응용프로그램은 다음과 같은 코드 프래그먼트를 사용하는 저장 프로시듀어를 호출합니다.

```
...
CallableStatement stpCall ;
String sql = "CALL PARTS_ON_HAND ( ?,?,? )" ;
stpCall = con.prepareCall( sql ) ; /* con is the connection */
stpCall.setInt( 1, variable1 ) ;
stpCall.setBigDecimal( 2, variable2 ) ;
stpCall.setInt( 3, variable3 ) ;
stpCall.registerOutParameter( 2, Types.DECIMAL, 2 ) ;
stpCall.registerOutParameter( 3, Types.INTEGER ) ;
stpCall.execute() ;
variable2 = stpCall.getBigDecimal(2) ;
variable3 = stpCall.getInt(3) ;
...
```

외부 이름 'parts!onhand'를 갖는 CALL문에 지정된 프로시듀어 이름을 데이터베이스에서 찾게 되면 응용프로그램 코드는 클래스 부분에서 Java 방법 *onhand*를 호출합니다.

CLOSE

CLOSE문은 커서를 닫습니다. 커서가 열릴 때 결과 테이블이 작성되면, 그 테이블은 소멸됩니다.

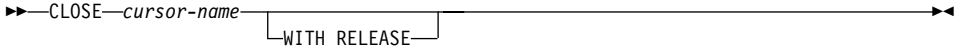
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

필요 사항 없음. 커서를 사용할 때 필요한 권한에 대해서는 952 페이지의 『DECLARE CURSOR』에서 참조하십시오.

구문



설명

커서 이름

닫을 커서를 식별합니다. *커서 이름*은 선언된 커서를 DECLARE CURSOR 문에 설명된 대로 식별해야 합니다. CLOSE문이 실행될 때, 커서는 열린 상태여야 합니다.

WITH RELEASE

커서에 대해 보유된 모든 읽기 잠금의 해제가 시도됩니다. 모든 읽기 잠금이 꼭 해제될 필요는 없다는 것을 유념하십시오. 이들 잠금은 다른 조작이나 활동에 대해서도 보유될 지 모릅니다.

주

- 작업 단위의 끝에서, 응용프로그램 프로세스에 속하고 WITH HOLD 옵션없이 선언된 모든 커서는 내재적으로 닫힙니다.
- CLOSE는 확약이나 구간 복원 조작을 야기하지 않습니다.

- WITH RELEASE절은 분리 레벨 CD나 UR하에서 작동되는 커서에 대해 영향을 미치지 않습니다. 분리 레벨 RS나 RR하에서 작동되는 커서에 대해 지정된 경우, WITH RELEASE는 그 분리 레벨의 보증 중 일부를 종료합니다. 특히, 그 커서가 다시 열리면, RS 커서에 '반복 불가능한 읽기' 현상이 발생할 수 있으며, RR 커서에는 '반복 불가능한 읽기' 또는 '가상' 현상이 발생할 수도 있습니다. 1449 페이지의 『부록I. 분리 레벨 비교』에서 자세한 내용을 참조하십시오.

  원래 RR 또는 RS인 커서가 WITH RELEASE절을 사용하여 닫힌 후에 다시 열린 경우, 새로운 읽기 잠금이 확보됩니다.

- 호출 프로그램으로 리턴되기 전에 닫히지 않은 저장 프로시듀어 내에 있는 커서에 특별한 규칙이 적용됩니다. 583 페이지의 『주』에서 좀더 자세한 정보를 참조하십시오.

예

커서는 한번에 한 행을 C 프로그램 변수 dnum, dname 및 mnum로 페치할 때 사용됩니다. 마지막으로, 커서가 닫힙니다. 커서가 다시 열리면, 페치될 행들의 맨 앞에 다시 위치됩니다.

```
EXEC SQL  DECLARE C1 CURSOR FOR
    SELECT DEPTNO, DEPTNAME, MGRNO
    FROM TDEPT
    WHERE ADMRDEPT = 'A00';
EXEC SQL  OPEN C1;
 while (SQLCODE==0) {                                              .
    EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
       .
       .
 }
EXEC SQL  CLOSE C1;
```

COMMENT ON

COMMENT ON문은 다양한 오브젝트의 카탈로그 설명에 주석을 추가하거나 대체합니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

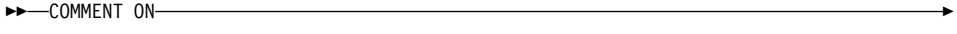
권한 부여

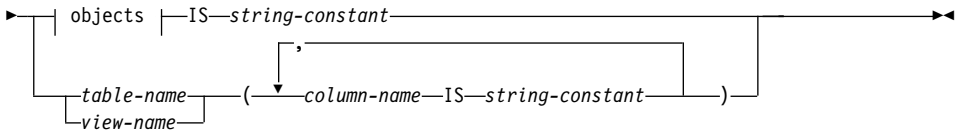
COMMENT ON문의 권한 부여 ID가 보유한 특권에는 다음 중 적어도 하나가 포함되어야 합니다.

- SYSADM 또는 DBADM
- 오브젝트에 대한 카탈로그 뷰(view)의 DEFINER 컬럼(스키마에 대한 OWNER 컬럼)에 레코드되는 오브젝트의 정의자(컬럼이나 제한조건의 기저 테이블)
- 스키마에 대한 ALTERIN 특권(두 개 이상의 부분으로 구성되는 이름을 허용하는 오브젝트에만 적용 가능함)
- 오브젝트에 대한 CONTROL 특권(색인, 패키지, 테이블 및 뷰 오브젝트에만 적용 가능함)
- 오브젝트에 대한 ALTER 특권(테이블 오브젝트에만 적용 가능함)

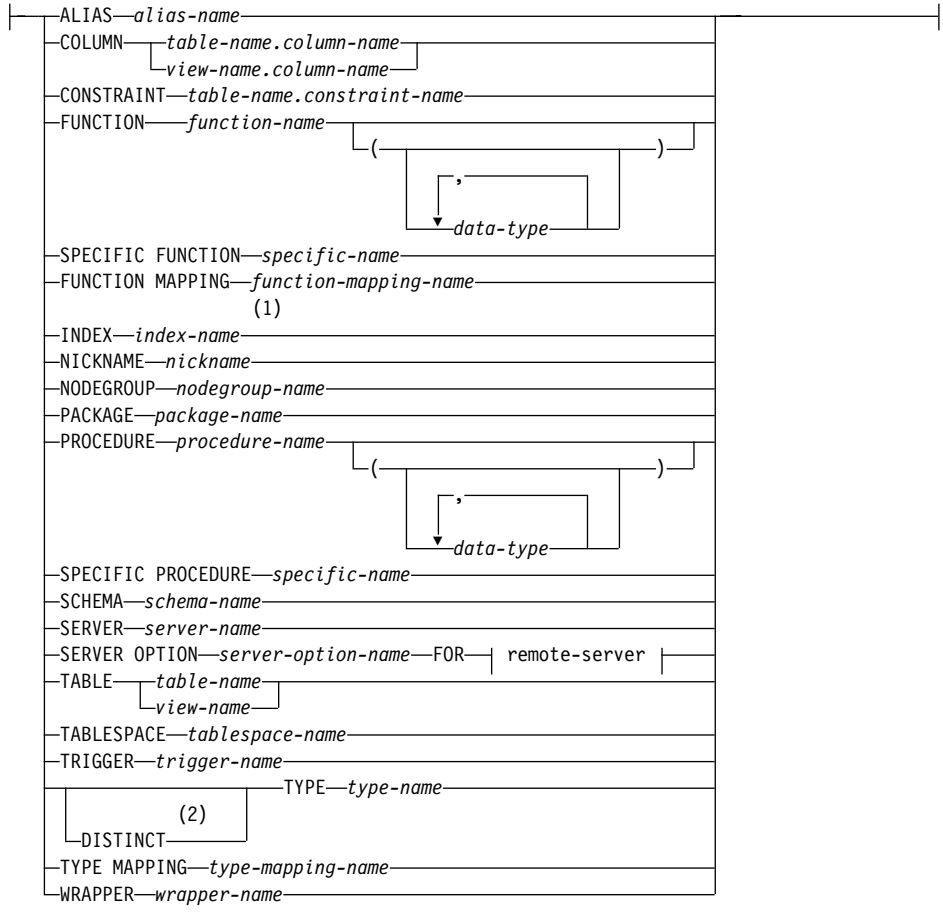
테이블 공간이나 권한 부여 ID의 경우, SYSADM 또는 SYSCTRL 권한이 있어야 합니다.

구문

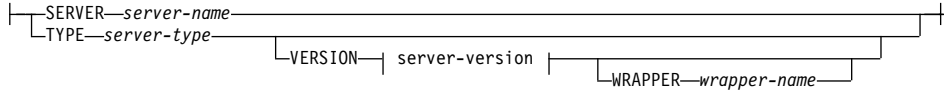




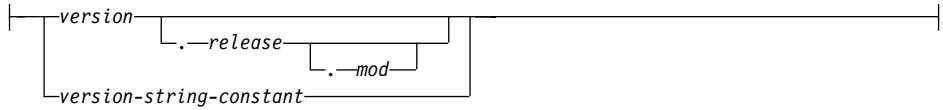
objects:



remote-server:



server-version:



주:

1 *색인 이름*은 색인 또는 색인 스펙의 이름일 수 있습니다.

2 키워드 DATA는 DISTINCT의 동의어로 사용할 수 있습니다.

설명

ALIAS *별명*

별명에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *별명*은 카탈로그에서 설명되는 별명을 식별해야 합니다(SQLSTATE 42704). 주석은 별명을 설명하는 행에 대한 SYSCAT.TABLES의 REMARKS 컬럼 값을 대체합니다.

COLUMN *테이블 이름.컬럼 이름* 또는 *뷰 이름.컬럼 이름*

컬럼에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *테이블 이름.컬럼 이름* 또는 *뷰 이름.컬럼 이름* 조합은 카탈로그에 설명되어 있는 컬럼과 테이블 조합을 식별해야 합니다(SQLSTATE 42704). 주석은 컬럼을 설명하는 행에 대한 SYSCAT.COLUMNS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

주석은 작동 불능 뷰에 대해서는 작성할 수 없습니다(SQLSTATE 51024).

CONSTRAINT *테이블 이름.제한조건 이름*

제한조건에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *테이블 이름.제한조건 이름* 조합은 카탈로그에 설명되어 있는 제한조건과, 그 제한조건이 제한하는 테이블을 식별해야 합니다(SQLSTATE 42704). 주석은 제한조건을 설명하는 행에 대한 SYSCAT.TABCONST 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

FUNCTION

함수에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. 지정된 함수 인스턴스는 카탈로그에 기술되어 있는 사용자 정의 함수 또는 함수 템플리트어야 합니다.

함수 인스턴스를 식별하는 데 사용할 수 있는 방법이 여러 가지 있습니다.

FUNCTION *함수 이름*

특수한 함수를 식별하고, *함수 이름*을 갖는 함수가 정확히 하나가 있는 경우에만 유효합니다. 그러므로 식별되는 함수는 이에 대해 정의된 임의 수의 매개변수를 가질 수 있습니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 명명되거나 암시적인 스키마에 이 이름을 가진 함수가 없으면, 오류(SQLSTATE 42704)가 발생합니다. 명명된 또는 암시적인 스키마에 하나 이상의 특정 함수 인스턴스가 있는 경우, 오류(SQLSTATE 42854)가 발생합니다.

FUNCTION *함수 이름*(*데이터 유형*,...)

주석을 지정할 함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 함수 선택 알고리즘은 사용되지 *않습니다*.

함수 이름

주석을 정의할 함수의 이름을 제공합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

(*데이터 유형*,...)

해당 위치에 있는 CREATE FUNCTION문에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형 수, 그리고 데이터 유형의 논리적 병합은 주석을 추가하거나 대체할 특정 함수를 식별하는 데 사용됩니다.

*데이터 유형*이 규정되어 있지 않으면, SQL 경로의 스키마를 검색하여 유형 이름을 해석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대한 정밀도 또는 스케일, 길이를 지정하는 것은 필요하지 않습니다. 데이터 유형 일치사항을 찾을 때 이 속성들이 무시됨을 나타내기 위해 빈 괄호 집합 대신 코딩될 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용될 수 없습니다(SQLSTATE 42601).

그러나, 길이, 정밀도 또는 스케일을 적어넣는 경우, 그 값은 CREATE FUNCTION문에 지정된 값과 정확히 일치해야 합니다.

0 <n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치(Matching)는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

(FOR BIT DATA 속성은 대조용 시그니처의 부분에 대해 고려되지 않습니다. 그러므로, 예를 들어 이름에 지정된 CHAR FOR BIT DATA가 CHAR에 대해서만 정의된 함수와 일치하지 않습니다.)

지정된 시그니처의 어떤 함수도 명명되거나 암시된 스키마에 존재하지 않을 경우, 오류(SQLSTATE 42883)가 발생합니다.

SPECIFIC FUNCTION *특정 이름*

주석이 함수에 대해 추가되거나 대체됨을 나타냅니다(함수를 식별하는 다른 방법에 대해서는 FUNCTION을 참조하십시오.). 함수 작성시 지정되거나 기본값인 특정 이름을 사용하여 주석이 붙여질 특수한 사용자 정의 함수를 식별합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. *특정 이름*은 명명된 또는 암시된 스키마에서 특정 함수 인스턴스를 식별합니다. 그렇지 않은 경우, 오류(SQLSTATE 42704)가 발생합니다.

SYSIBM 스키마나 SYSFUN 스키마 중 하나에 속한 함수에는 주석을 넣을 수 없습니다(SQLSTATE 42832).

주석은 함수를 설명하는 행에 대한 SYSCAT.FUNCTIONS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

FUNCTION MAPPING *함수 맵핑 이름*

함수 맵핑에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *함수 맵핑 이*

름은 카탈로그에 기술되어 있는 함수 맵핑을 식별해야 합니다(SQLSTATE 42704). 주석은 함수 맵핑을 설명하는 행에 대한 SYSCAT.FUNCMAPPINGS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

INDEX *색인 이름*

색인이나 색인 스펙을 위한 주석이 추가 또는 대체될 것임을 나타냅니다. *색인 이름*은 카탈로그에서 기술되는 구별 색인이나 색인 스펙을 식별해야 합니다(SQLSTATE 42704). 주석은 색인 또는 색인 스펙을 기술하는 행에 대한 SYSCAT.INDEXES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

NICKNAME *별명*

별명에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *별명*은 카탈로그에서 기술되는 별명이어야 합니다(SQLSTATE 42704). 주석은 별명을 설명하는 행에 대한 SYSCAT.TABLES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

NODEGROUP 노드 *그룹 이름*

노드 그룹에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. 노드 *그룹 이름*은 카탈로그에 기술된 명확한 노드 그룹을 식별해야 합니다(SQLSTATE 42704). 주석은 노드 그룹을 기술하는 행에 대한 SYSCAT.NODEGROUPS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

PACKAGE *패키지 이름*

패키지에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *패키지 이름*은 카탈로그에 설명된 명백한 패키지를 식별해야 합니다(SQLSTATE 42704). 주석은 패키지를 설명하는 행에 대한 SYSCAT.PACKAGES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

PROCEDURE

프로시듀어에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. 지정된 프로시듀어 인스턴스는 카탈로그에 기술되어 있는 저장 프로시듀어야 합니다.

프로시듀어 인스턴스를 식별하는 데 사용할 수 있는 방법이 여러 가지 있습니다.

PROCEDURE *프로시듀어 이름*

특정 프로시듀어를 식별하며, 스키마에 있는 *프로시듀어 이름*을 갖는 프로

시듀어가 하나 뿐인 경우에만 유효합니다. 그러므로 식별되는 프로시듀어에 대해서는 수에 관계없이 매개변수를 정의할 수 있습니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 명명되거나 내재된 스키마에 이 이름의 프로시듀어가 존재하지 않는 경우, 오류(SQLSTATE 42704)가 발생합니다. 명명된 스키마나 내재된 스키마에 프로시듀어의 특정 인스턴스가 하나 이상 있을 경우, 오류(SQLSTATE 42854)가 발생합니다.

PROCEDURE *프로시듀어 이름* (*데이터 유형*,...)

이것은 주석을 정의할 프로시듀어를 고유하게 식별하는 함수 시그니처를 제공하는 데 사용됩니다.

프로시듀어 이름

주석이 지정될 프로시듀어의 프로시듀어 이름을 제공합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

(*데이터 유형*,...)

해당 위치에 있는 CREATE PROCEDURE문에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형의 수와 데이터 유형의 논리적 병합은 주석을 추가하거나 대체할 특정 프로시듀어를 식별하는 데 사용됩니다.

*데이터 유형*이 규정되어 있지 않으면, SQL 경로의 스키마를 검색하여 유형 이름을 해석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대한 정밀도 또는 스케일, 길이를 지정하는 것은 필요하지 않습니다. 데이터 유형 일치사항을 찾을 때 이 속성들이 무시됨을 나타내기 위해 빈 괄호 집합 대신 코딩될 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용될 수 없습니다(SQLSTATE 42601).

그러나, 길이, 정밀도 또는 스케일이 코드화된 경우, 그 값은 CREATE PROCEDURE문에 지정된 것과 정확히 일치해야 합니다.

0 <n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치(Matching)는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 시그니처의 프로시듀어가 명명된 스키마나 내재된 스키마에 존재하지 않을 경우, 오류(SQLSTATE 42883)가 발생합니다.

SPECIFIC PROCEDURE *특정 이름*

프로시듀어에 대해 주석이 추가되거나 대체될 것임을 나타냅니다(프로시듀어를 식별하는 다른 방법에 대해서는 PROCEDURE를 참조하십시오.). 프로시듀어 작성시 지정되거나 기본 설정된 특정 이름을 사용하여 주석이 붙여질 특정 저장 프로시듀어를 식별합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. *특정 이름*은 명명된 스키마 또는 내재된 스키마의 특정 프로시듀어 인스턴스를 식별해야 합니다. 그렇지 않으면, 오류(SQLSTATE 42704)가 발생합니다.

주석은 프로시듀어를 기술하는 행에 대한 SYSCAT.PROCEDURES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

SCHEMA *스키마 이름*

스카마에 대해 주석이 추가되거나 대체될 것임을 나타냅니다. *스키마 이름*은 카탈로그에 기술된 스키마를 식별해야 합니다(SQLSTATE 42704). 주석은 스키마를 기술하는 행에 대한 SYSCAT.SCHEMATA 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

SERVER *서버 이름*

데이터 소스에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *서버 이름*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다(SQLSTATE 42704). 주석은 데이터 소스를 설명하는 행에 대한 SYSCAT.SERVERS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

SERVER OPTION *서버 옵션 이름* **FOR** *원격 서버*

서버 옵션에 대한 주석이 추가되거나 대체될 것임을 나타냅니다.

서버 옵션 이름

서버 옵션을 식별합니다. 이 옵션은 카탈로그에서 기술되는 것이어야 합니다(SQLSTATE 42704). 주석은 서버 옵션을 기술하는 행에 대한 SYSCAT.SERVEROPTIONS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

원격 서버

*서버 옵션*이 적용하는 데이터 소스를 기술합니다.

SERVER *서버 이름*

*서버 옵션*이 적용하는 데이터 소스에 이름을 부여합니다. *서버 이름*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다.

TYPE *서버 유형*

*서버 옵션*이 적용되는 데이터 소스의 유형을 지정합니다(예를 들어, OS/390이나 Oracle의 경우 DB2 Universal Database). *서버 유형*은 대문자나 소문자 어느 것으로든 지정될 수 있습니다. 카탈로그에는 대문자로 보관될 것입니다.

VERSION

*서버 이름*으로 식별된 데이터 소스의 버전을 지정합니다.

버전

버전 번호를 지정합니다. *버전*은 정수여야 합니다.

릴리스

*버전*으로 표시된 버전의 릴리스 번호를 지정합니다. *릴리스*는 정수여야 합니다.

mod

*릴리스*로 표시된 릴리스의 수정 번호를 지정합니다. *mod*는 정수여야 합니다.

버전 문자열 상수

완전한 버전 지정을 합니다. *버전 문자열 상수*는 단일 값(예를 들어, '8i')이거나 또는 *버전*, *릴리스* 및 *mod*의 결합된 값(예를 들어, '8.0.3')일 수 있습니다.

WRAPPER *랩퍼 이름*

*서버 이름*으로 참조되는 데이터 소스를 액세스하기 위해 사용되는 랩퍼를 식별합니다.

TABLE *테이블 이름* 또는 *뷰 이름*

테이블 또는 뷰에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *테이블 이름*이나 *뷰 이름*은 카탈로그에서 설명된 테이블이나 뷰(별명이 아님)를 식별해야 하고(SQLSTATE 42704) 선언된 임시 테이블은 식별하지 않아야 합니다(SQLSTATE 42995). 주석은 기본 테이블이나 뷰를 설명하는 행에 대한 SYSCAT.TABLES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

TABLESPACE *테이블 공간 이름*

테이블 공간에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *테이블 공간 이름*은 카탈로그에 설명된 명백한 테이블 공간을 식별해야 합니다(SQLSTATE 42704). 주석은 테이블 공간을 설명하는 행에 대한 SYSCAT.TABLESPACES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

TRIGGER *트리거 이름*

트리거에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *트리거 이름*은 카탈로그에 설명된 명백한 트리거를 식별해야 합니다(SQLSTATE 42704). 주석은 트리거를 설명하는 행에 대한 SYSCAT.TRIGGERS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

TYPE *유형 이름*

사용자 정의 유형에 대해 주석이 추가되거나 교체될 것임을 나타냅니다. *유형 이름*은 카탈로그에 기술된 사용자 정의 유형을 식별해야 합니다(SQLSTATE 42704). DISTINCT가 지정되는 경우 *유형 이름*은 카탈로그에 기술된 구별 유형을 식별해야 합니다(SQLSTATE 42704). 주석은 사용자 정의 유형을 기술하는 행에 대해 SYSCAT.DATATYPES 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

TYPE MAPPING *유형 맵핑 이름*

사용자 정의 데이터 유형 맵핑에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *유형 맵핑 이름*은 카탈로그에 기술되어 있는 데이터 유형 맵핑을 식별해야 합니다(SQLSTATE 42704). 주석은 맵핑을 설명하는 행에 대한 SYSCAT.TYPEMAPPINGS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

WRAPPER *랩퍼 이름*

랩퍼(wrapper)에 대한 주석이 추가되거나 대체될 것임을 나타냅니다. *랩퍼 이름*은 카탈로그에 기술된 랩퍼를 식별해야 합니다(SQLSTATE 42704). 주석은 랩퍼를 기술하는 행에 대한 SYSCAT.WRAPPERS 카탈로그 뷰의 REMARKS 컬럼 값을 대체합니다.

IS *문자열 상수*

추가되거나 대체될 주석을 지정합니다. *문자열 상수*는 최대 254바이트까지의 문자열 상수일 수 있습니다(캐리지 리턴(CR)과 라인 피드(LF)도 각각 1바이트로 계산됨).

테이블 이름|*뷰 이름* ({ *컬럼 이름* **IS** *문자열 상수* } **...**)

이 양식의 COMMENT ON문은 테이블이나 뷰(view)의 여러 컬럼에 대해 주석을 지정할 수 있는 기능을 제공합니다. 컬럼 이름은 규정하지 않아야 하고, 각 이름은 지정된 테이블이나 뷰의 컬럼을 식별해야 하며, 테이블이나 뷰는 카탈로그에서 설명되어야 합니다. *테이블 이름*은 선언된 임시 테이블이 될 수 없습니다(SQLSTATE 42995).

주석은 작동 불능 뷰의 컬럼에 대해서는 작성할 수 없습니다(SQLSTATE 51024).

예

예 1: EMPLOYEE 테이블에 대해 주석을 추가합니다.

```
COMMENT ON TABLE EMPLOYEE
 IS 'Reflects first quarter reorganization'
```

예 2: EMP_VIEW1 뷰에 대해 주석을 추가합니다.

```
COMMENT ON TABLE EMP_VIEW1
 IS 'View of the EMPLOYEE table without salary information'
```

예 3: EMPLOYEE 테이블의 EDLEVEL 컬럼에 대해 주석을 추가합니다.

```
COMMENT ON COLUMN EMPLOYEE.EDLEVEL
 IS 'highest grade level passed in school'
```

예 4: EMPLOYEE 테이블의 서로 다른 두 컬럼에 주석을 추가합니다.

```
COMMENT ON EMPLOYEE
  (WORKDEPT IS 'see DEPARTMENT table for names',
   EDLEVEL IS 'highest grade level passed in school' )
```

예 5: Pellow는 주석이 지정될 특정 함수를 식별하기 위한 시그니처를 사용하여 자신의 PELLOW 스키마에 작성한 CENTRE 함수에 대해 주석을 달고자 합니다.

```
COMMENT ON FUNCTION CENTRE (INT,FLOAT)
   IS 'Frank''s CENTRE fctn, uses Chebychev method'
```

예 6: McBride는 주석이 지정될 함수 인스턴스를 식별하기 위한 특정 이름을 사용하여 자신이 PELLOW 스키마에 작성한 또 다른 CENTRE 함수에 주석을 달고자 합니다.

```
COMMENT ON SPECIFIC FUNCTION PELLOW.FOCUS92 IS
 'Louise''s most triumphant CENTRE function, uses the
  Brownian fuzzy-focus technique'
```

예 7: CHEM 스키마의 함수 AUTOMIC_WEIGHT에 주석을 답니다. CHEM 스키마에는 이 이름을 갖는 함수가 하나뿐입니다.

```
COMMENT ON FUNCTION CHEM.ATOMIC_WEIGHT
   IS 'takes atomic nbr, gives atomic weight'
```

예 8: Eigler는 주석을 지정할 특정 프로시듀어를 식별하기 위한 시그니처를 사용하여 자신의 EIGLER 스키마에 작성한 SEARCH 프로시듀어에 대해 주석을 달고자 합니다.

```
COMMENT ON PROCEDURE SEARCH (CHAR,INT)
   IS 'Frank''s mass search and replace algorithm'
```

예 9: Macdonald는 주석을 지정할 프로시듀어 인스턴스를 식별하기 위한 특정 이름을 사용하여 EIGLER 스키마에 작성한 또 다른 SEARCH 함수에 주석을 달고자 합니다.

```
COMMENT ON SPECIFIC PROCEDURE EIGLER.DESTROY IS
   'Patrick''s mass search and destroy algorithm'
```

예 10: BIOLOGY 스키마의 프로시듀어 OSMOSIS에 대해 주석을 답니다. 이 스키마에는 이 이름을 갖는 프로시듀어가 하나뿐입니다.

```
COMMENT ON PROCEDURE BIOLOGY.OSMOSIS
   IS 'Calculations modelling osmosis'
```

예 11: INDEXSPEC이라는 이름의 색인 스펙을 주석처리하십시오.

```
COMMENT ON INDEX INDEXSPEC
   IS '최적화 알고리즘에게 별명 NICK1으로
   참조된 테이블에 색임이 있음을 나타내는 별명 스펙'
```

예 12: 기본 이름이 NET8인 랩퍼를 주석처리하십시오.

```
COMMENT ON WRAPPER NET8
   IS 'Oracle의 Net8 클라이언트 소프트웨어와 연관된 데이터 소스에 대한 랩퍼'
```

COMMIT

COMMIT문은 작업 단위(UOW)를 종료하고 그 작업 단위에 의해 수행된 데이터베이스 변경사항을 확약(commit)합니다.

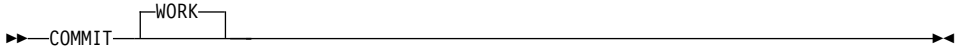
호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

필요 사항 없음.

구문



설명

COMMIT문이 실행되는 작업 단위는 종료되고 새로운 작업 단위가 시작됩니다. 작업 단위 중에 실행된 다음 명령문으로 작성된 모든 변경사항이 실행됩니다: ALTER, COMMENT ON, CREATE, DELETE, DROP, GRANT, INSERT, LOCK TABLE, REVOKE, SET INTEGRITY, SET 전이 변수 및 UPDATE.

그러나, 다음 명령문은 트랜잭션 제어하에 있지 않으며, 트랜잭션에 의한 변경은 COMMIT문을 실행하는 것과 무관합니다.

- SET CONNECTION,
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE,
- SET CURRENT EXPLAIN MODE,
- SET CURRENT EXPLAIN SNAPSHOT,
- SET CURRENT PACKAGESET,
- SET CURRENT QUERY OPTIMIZATION,
- SET CURRENT REFRESH AGE,

- SET EVENT MONITOR STATE,
- SET PASSTHRU,
- SET PATH,
- SET SCHEMA,
- SET SERVER OPTION.

WITH HOLD로 선언되어 열린 커서(cursor)에 필요한 잠금을 제외하고는, 초기 설정 이후의 작업 단위에 의해 확보된 모든 잠금은 해제됩니다. WITH HOLD로 정의되지 않은 모든 열린 커서들은 닫힙니다. WITH HOLD로 정의된 열기 커서는 열린 채 남아 있고, 그 커서는 결과 테이블의 다음 논리 행 이전에 위치됩니다. [63]모든 LOB 위치 지정자는 해제됩니다. 이것은 위치 지정자가 WITH HOLD 등록 정보를 갖는 커서를 통해 검색된 LOB 값과 연관될 때에도 성립합니다.

트랜잭션 내에 설정된 모든 저장 지점은 해제됩니다.

주

- 종료 전에 각 응용프로그램 프로세스가 명시적으로 작업 단위를 종료하는 것이 매우 바람직합니다. 응용프로그램이 COMMIT나 ROLLBACK문 없이 정상적으로 종료되면, 데이터베이스 관리 프로그램은 응용프로그램 환경에 따라 확약 또는 구간 복원을 수행합니다. 서로 다른 응용프로그램 환경에서 내재적으로 트랜잭션을 종료하는 방법에 대해서는 *응용프로그램 개발 안내서*에서 참조하십시오.
- 캐쉬된 동적 SQL문에서의 COMMIT 영향에 대해서는 1016 페이지의 『EXECUTE』에서 자세한 내용을 참조하십시오.
- 선언된 임시 테이블에서의 COMMIT 영향에 대해서는 958 페이지의 『DECLARE GLOBAL TEMPORARY TABLE』에서 자세한 내용을 참조하십시오.

예

최종 확약점 이후로 수행된 데이터베이스에 대해 변경사항을 확약합니다.

```
COMMIT WORK
```

63. FETCH는 위치된 UPDATE 또는 DELETE 명령문이 발행되기 전에 수행되어야 합니다.

복합 SQL(포함)

하나 이상의 다른 SQL문(*부속 명령문*)을 실행 가능한 블록에 결합합니다. SQL 프로시듀어 내의 복합 SQL문에 대해서는 1207 페이지의 『제7장 SQL 프로시듀어』에서 자세한 내용을 참조하십시오.

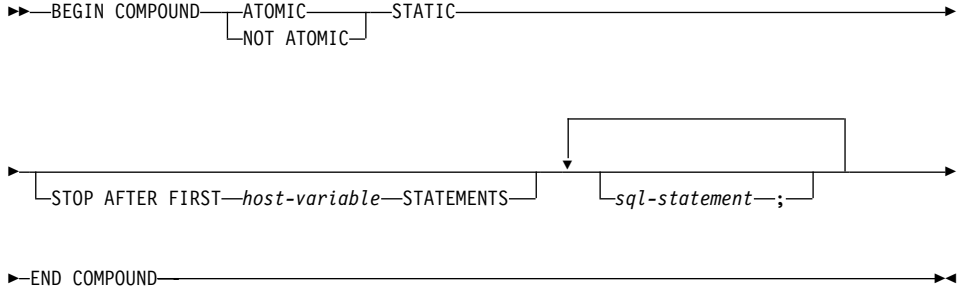
호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 전체 복합 SQL문 구조는 동적으로 준비될 수 없는 실행 가능한 명령문입니다. 이 명령문은 REXX에서 지원되지 않습니다.

권한 부여

복합 SQL문 그 자체에 대해서는 없습니다. 복합 SQL의 권한 부여 ID는 복합 SQL문 내에 포함된 모든 개별 명령문에 대해 적절한 권한을 갖고 있어야 합니다.

구문



설명

ATOMIC

복합 SQL문 내의 부속 명령문들 중 어느 하나가 실패할 경우, 성공적인 부속 명령문에 의해 수행된 변경사항을 포함하여, 부속 명령문 중 하나에 의해 데이터베이스에 대해 수행된 모든 변경사항이 실행 취소됨을 지정합니다.

NOT ATOMIC

부속 명령문의 실패 여부에 관계없이, 복합 SQL문에서는 다른 부속 명령문에서 데이터베이스를 변경한 내용을 복원하지 않음을 지정합니다.

STATIC

모든 부속 명령문에 대한 입력 변수가 원래의 값을 보유함을 지정합니다. 예를 들어, 다음 명령문 다음에

```
SELECT ... INTO :abc ...
```

다음 명령문이 오면,

```
UPDATE T1 SET C1 = 5 WHERE C2 = :abc
```

UPDATE문은 SELECT INTO 다음에 오는 값이 아니라, 복합 SQL문의 실행을 시작할 때 :abc가 가지고 있었던 값을 사용합니다.

두 개 이상의 부속 명령문에 의해 같은 변수가 설정되면, 복합 SQL문을 따르는 해당 변수의 값은 마지막 부속 명령문에 의해 설정되는 값입니다.

주: 비정적 동작은 지원되지 않습니다. 이 는 부속 명령문이 실행되는 대로 비순차적으로 열람되어야 하고, 상호 종속성을 갖고 있어서는 안된다는 것을 의미합니다.

STOP AFTER FIRST

특정 개수의 부속 명령문만 실행됨을 지정합니다.

호스트 변수

실행될 부속 명령문 개수를 지정하는 작은 정수(small integer).

STATEMENTS

STOP AFTER FIRST 호스트 변수절을 완료합니다.

sql문

다음을 제외하고 모든 실행 명령문이 embedded 정적 복합 SQL문 내에 포함될 수 있습니다.

CALL
CLOSE
CONNECT
복합 SQL
DESCRIBE
DISCONNECT
EXECUTE IMMEDIATE
FETCH
OPEN
PREPARE
RELEASE(연결)
RELEASE SAVEPOINT
ROLLBACK
SAVEPOINT
SET CONNECTION

COMMIT문이 포함되면, 이것은 마지막 부속 명령문이어야 합니다. COMMIT가 이 위치에 있으면, STOP AFTER FIRST *호스트 변수* STATEMENTS 절에 모든 부속 명령문이 실행되는 것은 아님을 나타낼 경우에도 이 명령문이 발행됩니다. 예를 들어, COMMIT가 100개의 부속 명령문으로 된 복합 SQL 블록에서 마지막 부속 명령문이라고 할 경우, STOP AFTER FIRST STATEMENTS 절에 50개의 부속 명령문만 실행되도록 표시되면 COMMIT는 51번째 부속 명령문입니다.

CONNECT 유형 2를 사용하거나 XA 분산 트랜잭션 프로세싱 환경에서 수행되고 있을 때 COMMIT를 포함시키면 오류가 리턴됩니다(SQLSTATE 25000).

규칙

- DB2 Connect는 복합 SQL 블록에서 LOB 컬럼을 선택하는 SELECT문을 지원하지 않습니다.
- 복합 SQL문 내에서는 어떠한 호스트 언어도 코딩할 수 없습니다. 즉, 복합 SQL문을 구성하는 부속 명령문들 사이에 호스트 언어 코드를 사용할 수 없습니다.
- NOT ATOMIC Compound SQL 명령문만 DB2 Connect에서 승인됩니다.
- 복합 SQL문은 중첩될 수 없습니다.
- 최소단위 복합 SQL문은 저장 지점 내에서 발행될 수 없습니다(SQLSTATE 3B002).

주

전체 복합 SQL문에 대해 하나의 SQLCA가 리턴됩니다. 그 SQLCA에 있는 대부분의 정보는 마지막 부속 명령문을 처리할 때 응용프로그램 서버(AS)가 설정한 값을 반영합니다. 예를 들어,

- SQLCODE 및 SQLSTATE는 보통 마지막 부속 명령문에 대한 것입니다(예외는 다음에서 설명되어 있음).

- '데이터가 없음' 경고(SQLSTATE '02000')가 리턴될 경우, 그 경고에는 WHENEVER NOT FOUND 예외가 작동될 수 있는 순서에서 다른 경고보다 우선되는 순위가 주어집니다.[64]
- SQLWARN 표시기는 모든 부속 명령문에 대한 표시기 세트의 누적입니다.

최소 단위가 아닌 복합 SQL문 실행시 하나 이상의 오류가 발생하고 이들 중 어떤 것도 심각한 오류가 아니면, SQLERRMC에는 최대 7개까지의 이들 오류에 대한 정보가 포함됩니다. SQLERRMC의 첫번째 토큰은 발생한 총 오류 수를 나타냅니다. 나머지 토큰에는 서수 위치와 복합 SQL문 내의 실패한 부속 명령문의 SQLSTATE가 포함됩니다. 형식은 다음 양식의 문자열입니다.

```
nnnXsssccccc
```

여기서, 7번까지 반복하는 X로 시작하는 부속문자열이 있고 문자열 요소는 다음과 같이 정의됩니다.

nnn 오류를 생성한 총 명령문 수.[65]이 필드는 왼쪽으로 정렬되고 빈칸(blank)으로 채워집니다.

X 토큰 분리자 X'FF'.

sss 오류를 야기한 명령문의 서수 위치.[65]예를 들어, 첫번째 명령문이 실패하면, 이 필드에는 ' 1 '이 포함됩니다.

ccccc 오류의 SQLSTATE.

두 번째 SQLERRD 필드에는 실패한 명령문 수가 포함됩니다(음수의 SQLCODE 리턴).

SQLCA의 세번째 SQLERRD 필드는 모든 부속 명령문에 의해 영향을 받는 행 수의 누적입니다.

---

64. 즉, 결국 응용프로그램에 리턴되는 SQLCODE, SQLERRML, SQLERRMC, SQLERRP는 '데이터 없음'을 트리거한 부속 명령문의 SQLCODE, SQLERRML, SQLERRMC, SQLERRP입니다. 복합 SQL문 내에 하나 이상의 '데이터 없음' 경고가 있으면, 마지막 부속 명령문에 대한 필드는 리턴되는 필드입니다.

65. 번호가 999를 초과할 경우, 0부터 다시 계수가 시작됩니다.

SQLCA의 네번째 SQLERRD 필드는 성공한 부속 명령문 수입니다. 예를 들면, 복합 SQL문의 세번째 부속 명령문이 실패하면, 네번째 SQLERRD 필드는 2로 설정됩니다. 이것은 두 개의 부속 명령문이 오류 발생 전에 성공적으로 처리되었음을 의미합니다.

SQLCA의 다섯번째 SQLERRD 필드는 제한조건 활동을 트리거한 모든 부속 명령문에 대한 참조 무결성 제한조건의 실행으로 갱신되거나 삭제된 행 수의 누적 값입니다.

예

예 1: C 프로그램에서, ACCOUNTS와 TELLERS 테이블을 모두 갱신하는 복합 SQL문을 발행합니다. 명령문 중에 오류가 있으면, 모든 명령문의 효과를 복원합니다. 오류가 없으면, 현재 작업 단위를 확약합니다.

```
EXEC SQL BEGIN COMPOUND ATOMIC STATIC
  UPDATE ACCOUNTS SET ABALANCE = ABALANCE + :delta
    WHERE AID = :aid;
  UPDATE TELLERS SET TBALANCE = TBALANCE + :delta
    WHERE TID = :tid;
  INSERT INTO TELLERS (TID, BID, TBALANCE)  VALUES (:i, :branch_id, 0);
  COMMIT;
 END COMPOUND;
```

예 2: C 프로그램에서, 10개의 데이터 행을 데이터베이스에 삽입합니다. 호스트 변수 :nbr에 값 10이 있고 S1이 준비된 INSERT문이라고 가정합시다. 또한, 오류에 관계없이 모든 삽입이 시도된다고 가정합니다(NOT ATOMIC).

```
EXEC SQL BEGIN COMPOUND NOT ATOMIC STATIC STOP AFTER FIRST :nbr STATEMENTS
    EXECUTE S1 USING DESCRIPTOR :*sqlda0;
    EXECUTE S1 USING DESCRIPTOR :*sqlda1;
    EXECUTE S1 USING DESCRIPTOR :*sqlda2;
    EXECUTE S1 USING DESCRIPTOR :*sqlda3;
    EXECUTE S1 USING DESCRIPTOR :*sqlda4;
    EXECUTE S1 USING DESCRIPTOR :*sqlda5;
    EXECUTE S1 USING DESCRIPTOR :*sqlda6;
    EXECUTE S1 USING DESCRIPTOR :*sqlda7;
    EXECUTE S1 USING DESCRIPTOR :*sqlda8;
    EXECUTE S1 USING DESCRIPTOR :*sqlda9;
END COMPOUND;
```

CONNECT(유형 1)

CONNECT(유형 1)문은 응용프로그램 프로세스를 원격 작업 단위(UOW)에 대한 규칙에 따라 식별된 응용프로그램 서버(AS)에 연결합니다.

응용프로그램 프로세스는 한번에 하나의 응용프로그램에만 연결될 수 있습니다. 이것을 *현재 서버*라고 합니다. 기본 응용프로그램 서버(AS)는 응용프로그램 리퀘스터가 초기설정될 때 설정됩니다. 내재적 연결이 사용 가능하고 응용프로그램 프로세스가 시작된 경우, 이것은 기본 응용프로그램 서버(AS)에 내재적으로 연결됩니다. 응용프로그램 프로세스는 CONNECT TO문을 발행하여 다른 응용프로그램 서버(AS)에 명시적으로 연결할 수 있습니다. 연결은 CONNECT RESET문 또는 DISCONNECT문이 발행되거나, 다른 CONNECT TO문이 응용프로그램 서버를 변경할 때까지 계속됩니다.

연결 상태에 대한 개념과 37 페이지의 『원격 작업 단위 연결 관리』에서 자세한 내용을 참조하십시오. CONNECT 동작에 대한 프레임 워크를 판별하는 사전 처리 컴파일러에 대해서는 45 페이지의 『분산 작업 단위(DUOW) 의미 정의 옵션』에서 참조하십시오.

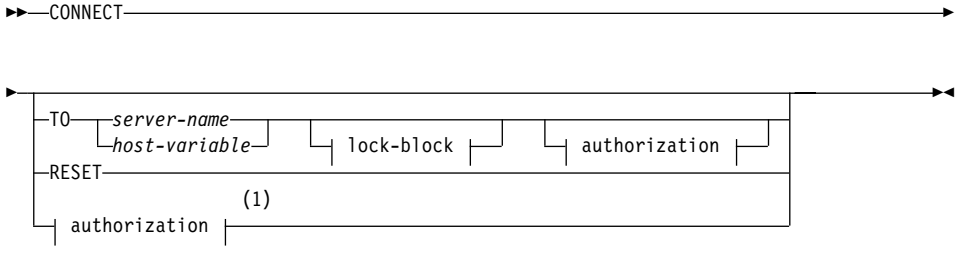
호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

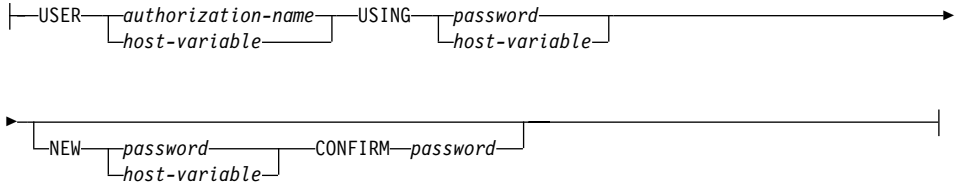
권한 부여

명령문의 권한 부여 ID는 식별된 응용프로그램 서버(AS)에 연결할 수 있는 권한이 부여되어야 합니다. 데이터베이스에 대한 인증 설정값에 따라, 클라이언트나 서버에서 권한 점검이 수행될 수 있습니다. 파티션된 데이터베이스의 경우, 사용자와 그룹 정의는 파티션이나 노드 전반에 걸쳐 동일해야 합니다. 인증 설정에 대한 정보에 대해서는 *관리 안내서* AUTHENTICATION 데이터베이스 관리 프로그램 구성 매개변수를 참조하십시오.

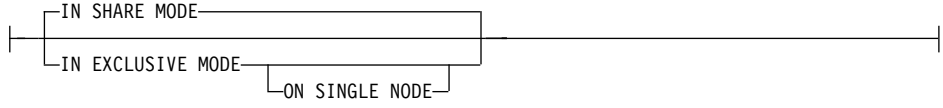
구문



authorization:



lock-block:



주:

1 이 양식은 내재적 연결이 가능한 경우에만 유효합니다.

설명

CONNECT (피연산자 없음)

현재 서버에 대한 정보를 리턴합니다. "성공적인 연결"에 기술된 것과 같이 SQLCA의 SQLERRP 필드에 정보가 리턴됩니다.

연결 상태가 존재하면, 권한 부여 ID와 데이터베이스 별명이 SQLCA의 SQLERRMC 필드에 위치됩니다. 권한 부여 ID가 8 바이트보다 길 경우, 그 ID는 8 바이트로 절단되고, 그 절단은 SQLCA의 SQLWARN0 및 SQLWARN1 필드에서 각각 'W' 및 'A'를 사용하여 플래그 처리됩니다. 데이

터베이스 구성 매개변수 DYN_QUERY_MGMT가 사용 가능하면, SQLCA의 SQLWARN0 및 SQLWARN7 필드는 'W' 및 'E'를 각각 사용하여 플래그 처리됩니다.

어떤 연결도 존재하지 않고 내재적 연결이 가능하면, 내재적 연결이 수행됩니다. 내재적 연결이 사용 가능하지 않으면, 오류가 발생합니다(기존의 연결은 없음). 연 결이 없으면 SQLERRMC 필드는 공백입니다.

응용프로그램 서버의 국가 코드와 코드 페이지는 SQLERRMC 필드에 위치합니다(이들이 CONNECT TO문을 정상적으로 수행시).

이 양식의 CONNECT가 수행될 때,

- 응용프로그램 프로세스가 연결 가능한 상태에 있을 필요가 없습니다.
- 연결되면, 연결 상태가 변경되지 않습니다.
- 연결되지 않고 내재적 연결이 가능한 경우, 기본 응용프로그램 서버(AS)에 대한 연결이 만들어집니다. 이러한 경우, 응용프로그램 서버(AS)의 국가 코드와 코드 페이지는 성공한 CONNECT TO문과 같습니다.
- 연결되지 않고 내재적 연결이 가능하지 않으면, 응용프로그램 프로세스는 연결되지 않은 상태로 남아 있습니다.
- 커서가 닫히지 않습니다.

TO *서버 이름* 또는 *호스트 이름*

서버 이름이 들어 있는 *호스트 변수* 또는 지정된 *서버 이름*으로 응용프로그램 서버(AS)를 식별합니다.

*호스트 변수*가 지정되면, 길이가 8자 이하인 문자열 변수이어야 하며 표시기 변수를 포함해서는 안됩니다. *호스트 변수* 내에 들어 있는 *서버 이름*은 왼쪽으로 정렬되어야 하고 따옴표로 구분해서는 안 됩니다.

*서버 이름*은 응용프로그램 서버(AS)를 식별하는 데이터베이스 별명임에 유의하십시오. 응용프로그램 리퀘스터의 지역 디렉토리에 나열되어야 합니다.

주: MVS용 DB2는 16바이트의 위치 이름(location-name)을 지원하고 SQL/DS와 DB2/400 둘다는 18바이트 목표 데이터베이스 이름을 지원합니다. DB2 버전 7에서만 SQL CONNECT문에 8바이트의 데이터베이스

별명을 사용할 수 있습니다. 그러나, 데이터베이스 연결 서비스 디렉토리를 통해 18바이트의 데이터베이스 이름에 대응될 수 있습니다.

CONNECT TO문이 실행될 때, 응용프로그램 프로세스는 연결 가능한 상태여야 합니다(유형 1 CONNECT의 연결 상태에 대해서는 37 페이지의 『원격 작업 단위 연결 관리』 참조).

성공적인 연결:

CONNECT TO문이 성공하면,

- 모든 열린 커서가 닫히고, 모든 준비된 명령문이 파괴되며, 모든 잠금이 이전 응용프로그램 서버로부터 해제됩니다.
- 존재하는 경우 응용프로그램 프로세스는 이전 응용프로그램 서버(AS)로부터 분리되고 식별된 응용프로그램 서버(AS)에 연결됩니다.
- 응용프로그램 서버(AS)의 실제 이름(별명이 아님)은 CURRENT SERVER 특수 레지스터에 위치됩니다.
- 응용프로그램 서버(AS)에 대한 정보는 SQLCA의 SQLERRP 필드에 위치됩니다. 응용프로그램 서버(AS)가 IBM 제품이면, 정보는 *pppvvrrm* 양식으로 되어 있습니다. 여기서,
  - *ppp*는 다음과 같은 제품을 식별합니다.
    - MVS용 DB2에 대한 DSN
    - SQL/DS용 ARI
    - DB2/400용 QSQ
    - DB2 Universal Database용 SQL
  - *vv*는 '02'와 같은 두 자리의 버전 식별자입니다.
  - *rr*은 '01'과 같은 두 자리의 릴리스 식별자입니다.
  - *m*은 '0'과 같은 한 자리의 수정 레벨 식별자입니다.

  예를 들어, 응용프로그램 서버(AS)가 OS/2용 DB2 버전 1 릴리스 1일 경우, SQLERRP의 값은 'sQL01010'입니다. [66]

66. DB2 Universal Database 버전 7의 릴리스는 'sQL07010'입니다.

- SQLCA의 SQLERRMC 필드는 다음 값(X'FF'로 구분됨)으로 설정됩니다.
  1. 응용프로그램 서버의 국가 코드(또는 DDCS를 사용할 경우 공백)
  2. 응용프로그램 서버의 코드 페이지(또는 DDCS를 사용할 경우 CCSID)
  3. 권한 부여 ID(처음부터 8 바이트까지만)
  4. 데이터베이스 별명
  5. 응용프로그램 서버의 플랫폼 유형. 현재 식별되는 값은 다음과 같습니다.

토큰	서버
QAS	AS/400용 DB2 Universal Database
QDB2	OS/390용 DB2 Universal Database
QDB2/2	OS/2용 DB2 Universal Database
QDB2/6000	AIX용 DB2 Universal Database
QDB2/HPUX	HP-UX용 DB2 Universal Database
QDB2/LINUX	Linux용 DB2 Universal Database
QDB2/NT	Windows NT용 DB2 Universal Database
QDB2/PTX	NUMA-Q용 DB2 Universal Database
QDB2/SCO	SCO UnixWare용 DB2 Universal Database
QDB2/SNI	Siemens Nixdorf용 DB2 Universal Database
QDB2/SUN	Solaris 운영 체제용 DB2 Universal Database
QDB2/Windows 95	Windows 95 또는 Windows 98용 DB2 Universal Database
QSQLDS/VM	VM용 DB2 서버

QSQLDS/VSE VSE용 DB2 서버

6. 에이전트 ID. 응용프로그램 대신 데이터베이스 관리 프로그램 내에서 실행하는 대행자를 식별합니다. 이 필드는 데이터베이스 모니터에 의해 리턴되는 agent_id 요소와 같습니다.
7. 대행자 색인. 대행자의 색인을 식별하고 서비스에 사용됩니다.
8. 파티션 번호. 비파티션된 데이터베이스에 대해서 존재시, 그 값은 항상 0입니다.
9. 응용프로그램 클라이언트의 코드 페이지.
10. 파티션된 데이터베이스의 파티션 번호 데이터베이스를 파티션할 수 없는 경우, 값은 0입니다. 토큰은 버전 5 이상에 있습니다.

- SQLCA의 SQLERRD(1) 필드는 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로 변환시 예상되는 최대 차이를 혼합 문자 데이터(CHAR 데이터 유형)의 길이로 나타냅니다. 값 0 또는 1은 확장되지 않음을 나타냅니다. 1보다 큰 값은 가능한 확장값의 길이로 나타내며, 음수값은 가능한 축소값을 나타냅니다. [67]
- SQLCA의 SQLERRD(2) 필드는 데이터베이스 코드 페이지에서 응용프로그램 코드 페이지로 변환시 예상되는 최대 차이를 혼합 문자 데이터(CHAR 데이터 유형)의 길이로 나타냅니다. 값 0 또는 1은 확장되지 않음을 나타냅니다. 1보다 큰 값은 가능한 확장값의 길이로 나타내며, 음수값은 가능한 축소값을 나타냅니다. [67]
- SQLCA의 SQLERRD(3) 필드는 연결시 데이터베이스를 갱신할 수 있는 지를 나타냅니다. 데이터베이스는 처음에는 갱신 가능하지만, 작업 단위(UOW)에서 권한 부여 ID가 갱신을 수행할 수 없다고 판별할 경우 읽기 전용으로 변경됩니다. 값은 다음 중 하나입니다.
  - 1 - 갱신 가능
  - 2 - 읽기 전용
- SQLCA의 SQLERRD(4) 필드는 연결의 특정한 특성을 리턴합니다. 값은 다음 중 하나입니다.

67. 세부사항은 *응용프로그램 개발 안내서*에 있는 "복잡한 환경에서의 프로그래밍" 장의 "문자 변환 확장 인수" 절을 참조하십시오.

0 - N/A(1 단계 확약이면서 갱신자인 하위 레벨 클라이언트로부터 수행중인 경우에만 가능함).

1 - 1단계 확약

2 - 1단계 확약; 읽기 전용(TP 모니터 환경에서의 DRDA1 데이터베이스에 대한 연결에 대해서만 적용 가능함).

3 - 2단계 확약

- SQLCA의 SQLERRD(5) 필드는 연결의 인증 유형을 리턴합니다. 값은 다음 중 하나입니다.

  0 - 서버에 대해 인증됨

  1 - 클라이언트에 대해 인증됨

  2 - DB2 연결 사용에 인증됨

  3 - DCE 보안 사용에 인증됨

  255 - 지정되지 않은 인증.

  인증 유형에 대해서는 *관리 안내서*의 "데이터베이스 액세스 제어"에서 자세한 내용을 참조하십시오.

- SQLCA의 SQLERRD(6) 필드는 데이터베이스가 파티션될 때 연결된 파티션의 파티션 번호를 리턴합니다. 그렇지 않으면, 0의 값이 리턴됩니다.
- 성공적인 연결의 권한 부여 ID가 8 바이트보다 클 경우, SQLCA의 SQLWARN1 필드는 'A'로 설정됩니다. 이는 절단이 발생하였음을 나타냅니다. SQLCA의 SQLWARN0 필드는 이러한 경고를 나타내기 위해 'W'로 설정됩니다.
- 데이터베이스에 대한 데이터베이스 구성 매개변수 DYN_QUERY_MGMT가 사용 가능할 경우 SQLCA의 SQLWARN7 필드는 'E'로 설정됩니다. SQLCA의 SQLWARN0 필드는 이러한 경고를 나타내기 위해 'W'로 설정됩니다.

실패한 연결:

CONNECT TO문이 성공적이지 못한 경우:

- SQLCA의 SQLERRP 필드가 오류를 감지한 응용프로그램 리퀘스터에 있는 모듈의 이름으로 설정됩니다. 모듈 이름의 처음 세 자는 제품을 식별합

니다. 예를 들어, 응용프로그램 리퀘스터(AR)가 OS/2 데이터베이스 관리 프로그램 상에 있으면, 처음 세 자는 'sQL'입니다.

- 응용프로그램 프로세스가 연결 가능한 상태에 있지 않아서 CONNECT TO 문이 성공하지 못한 경우, 응용프로그램 프로세스의 연결 상태는 변경되지 않습니다.
- *서버 이름*이 지역 디렉토리에 나열되어 있지 않아서 CONNECT TO문이 성공하지 못한 경우, 오류 메시지(SQLSTATE 08001)가 발행되고 응용프로그램 프로세스의 연결 상태는 변경되지 않고 그대로 있습니다.
  - 응용프로그램 리퀘스터가 응용프로그램 서버(AS)에 연결되지 않았으면, 응용프로그램 프로세스는 연결되지 않고 그대로 있습니다.
  - 응용프로그램 리퀘스터가 응용프로그램 서버(AS)에 연결되었으면, 응용프로그램 프로세스는 그 응용프로그램 프로세스에 연결된 상태로 있습니다. 임의의 추가 명령문은 응용프로그램 서버(AS)에서 실행됩니다.
- 어떤 이유로 CONNECT TO문이 성공하지 못했으면, 응용프로그램 프로세스는 연결되지 않은 상태가 됩니다.

IN SHARE MODE

데이터베이스에 대한 또 다른 동시 연결을 허용하며, 다른 사용자가 독점 모드로 데이터베이스에 연결하지 못하도록 합니다.

IN EXCLUSIVE MODE [68]

독점 잠금을 보유하는 사용자와 같은 권한 부여 ID를 갖고 있지 않는 한, 동시에 응용프로그램 프로세스가 응용프로그램 서버(AS)에서 조작을 실행하는 것을 방지합니다.

ON SINGLE NODE

조정자 파티션이 독점 모드로 연결되도록 하고 다른 모든 파티션은 공유 모드로 연결되도록 지정합니다. 이 옵션은 파티션된 데이터베이스에서만 유효합니다.

RESET

현재 서버로부터 응용프로그램 프로세스를 단절합니다. 확약 조작이 수행됩니

68. 이 옵션은 DDCS에서 지원되지 않습니다.

다. 내재적 응용프로그램이 사용 가능하면, 응용프로그램 프로세스는 SQL문이 발행되지 않는 한 연결되지 않은 채로 남습니다.

USER *권한 부여 이름/호스트 변수*

응용프로그램 서버(AS)에 연결하려고 하는 사용자 ID를 식별합니다. 호스트 변수가 지정되면, 길이가 8자 이하인 문자열 변수이어야 하며, 표시기(indicator) 변수가 포함되어서는 안됩니다. 호스트 변수 내에 포함되는 사용자 ID는 왼쪽으로 정렬되어야 하고, 따옴표로 구분되어서는 안됩니다.

USING *암호/호스트 변수*

응용프로그램 서버에 연결하려고 하는 사용자 ID의 암호를 식별합니다. 암호 또는 호스트 변수는 18자까지 될 수 있습니다. 호스트 변수를 지정하면, 18바이트보다 크지 않는 길이 속성을 갖고 있어야 하고 표시기 변수를 포함하지 않는 문자열 변수여야 합니다.

NEW *함호/호스트 변수* **CONFIRM** *암호*

USER 옵션에 의해 식별되는 사용자 ID에게 할당되어야 하는 새로운 암호를 식별합니다. 암호 또는 호스트 변수는 18자까지 될 수 있습니다. 호스트 변수를 지정하면, 18바이트보다 크지 않는 길이 속성을 갖고 있어야 하고 표시기 변수를 포함하지 않는 문자열 변수여야 합니다. 암호가 변경될 시스템은 사용자 인증이 설정된 방법에 따라 다릅니다.

주

- 응용프로그램 프로세스에 의해 실행되는 첫번째 SQL문이 CONNECT TO문이 되도록 하는 것이 좋습니다.
- CONNECT TO문이 다른 사용자 ID와 암호를 갖는 현재 응용프로그램 서버에 대해 발행되면, 대화가 할당 취소되고 다시 할당됩니다. 데이터베이스 관리 프로그램에 의해 모든 커서가 닫힙니다(WITH HOLD 옵션이 사용된 경우 커서 위치가 유실됨).
- CONNECT TO문이 다른 사용자 ID와 암호를 갖는 현재 응용프로그램 서버에 대해 발행되면, 대화가 할당 취소되어 다시 할당되지 않습니다. 이 경우, 커서는 닫히지 않습니다.
- DB2 Universal Database Enterprise - Extended Edition을 사용하려면, 사용자나 응용프로그램이 `db2nodes.cfg` 파일(68 페이지의 『여러 파티션에 걸친 데

이터 파티션』에서 이 파일에 대한 자세한 정보 참조)에 나열된 파티션 중 하나에 연결해야 합니다. 모든 사용자가 동일한 파티션을 조정자 파티션으로 사용하는 일이 없도록 해야 합니다.

예

예 1: C 프로그램에서, 사용자 ID FERMAT와 암호 THEOREM을 사용하여 응용프로그램 서버(AS) TOROLAB3에 연결합니다. 여기서, TOROLAB3은 동일한 이름의 데이터베이스 별명입니다.

```
EXEC SQL  CONNECT TO TOROLAB3 USER FERMAT USING THEOREM;
```

예 2: C 프로그램에서, 해당되는 데이터베이스 별명이 호스트 변수 APP_SERVER (varchar(8))에 저장되는 응용프로그램 서버(AS)에 연결합니다. 성공적인 연결 다음에, 응용프로그램 서버(AS)의 3자 제품 식별자를 변수 PRODUCT(char(3))에 복사합니다.

```
EXEC SQL  CONNECT TO :APP_SERVER;
if (strncmp(SQLSTATE,'00000',5))
  strncpy(PRODUCT,sqlca.sqlerrp,3);
```

CONNECT(유형 2)

CONNECT(유형 2)문은 응용프로그램 프로세스를 식별된 응용프로그램 서버로 연결하고 응용프로그램이 지시하는 분산 작업 단위(DUOW)에 대한 규칙을 설정합니다. 그러면, 이 서버는 프로세스에 대한 현재 서버가 됩니다.

개념과 41 페이지의 『응용프로그램 직접 분산 작업 단위』에서 자세한 설명을 참조하십시오.

CONNECT(유형 1)문에서 대부분의 사항이 CONNECT(유형 2)문에도 적용됩니다. 여기서는 그 내용을 반복하지 않고, 유형 2의 요소 중에서 유형 1과 다른 부분에 대해서만 설명합니다.

호출

호출은 608 페이지의 『호출』과 같습니다.

권한 부여

권한은 608 페이지의 『권한 부여』와 같습니다.

구문

구문은 609 페이지의 『구문』과 같습니다. 유형 1과 2 사이의 선택은 사전 처리 컴파일러 옵션에 의해 판별됩니다. 이 옵션의 개요에 대해서는 45 페이지의 『분산 작업 단위(DUOW) 의미 정의 옵션』에서 참조하십시오. 자세한 사항은 *Command Reference*와 *Administrative API Reference*에서 제공됩니다.

설명

TO *서버 이름/호스트 변수*

서버 이름의 코딩 규칙은 유형 1과 같습니다.

SQLRULES(STD) 옵션이 유효하면, *서버 이름*은 응용프로그램 프로세스의 기본 연결을 식별하지 않아야 합니다. 그렇지 않으면 오류(SQLSTATE 08002)가 발생합니다.

SQLRULES(DB2) 옵션이 유효하고 *서버 이름*이 응용프로그램 프로세스의 기존 연결을 식별할 경우, 그 연결은 현재 연결이 되고, 기존 연결은 휴면

(dormant) 상태로 됩니다. 즉, 이 상황에서 CONNECT문을 사용하면 SET CONNECTION문을 사용한 경우와 동일한 결과가 나타납니다.

SQLRULES의 스펙에 관한 정보는 45 페이지의 『분산 작업 단위(DUOW) 의미 정의 옵션』에서 참조하십시오.

성공적인 연결:

CONNECT TO문이 성공하면,

- 응용프로그램 서버(AS)에 대한 연결이 작성되어(또는 비 휴면 상태가 되어) 현재가 된 후 상태가 보유됩니다.
- CONNECT TO가 현재 서버와 다른 서버로 지정되면, 현재 연결은 휴면 상태가 됩니다.
- CURRENT SERVER 특수 레지스터와 SQLCA는 CONNECT 유형 1과 같은 방법으로 갱신됩니다. 611 페이지를 참조하십시오.

실패한 연결:

CONNECT TO문이 성공적이지 못한 경우:

- 실패한 이유가 무엇이든지, 응용프로그램 프로세스의 연결 상태 및 연결 상태는 변경되지 않습니다.
- 실패한 CONNECT 유형 1에서와 마찬가지로, SQLCA의 SQLERRP 필드는 오류를 감지한 응용프로그램 리퀘스터 또는 서버에서 모듈 이름으로 설정됩니다.

CONNECT(피연산자 없음)**, IN SHARE/EXCLUSIVE MODE, USER** 및 **USING**

연결이 존재하면, 유형 2는 유형 2처럼 작동합니다. 권한 부여 ID와 데이터베이스 별명이 SQLCA의 SQLERRMC 필드에 위치됩니다. 연결이 존재하지 않을 경우, 내재적 연결에 대한 어떠한 시도도 수행되지 않고 SQLERRP 및 SQLERRMC 필드가 공백을 리턴합니다(응용프로그램은 이 필드를 점검하여 현재 연결이 존재하는 지를 점검할 수 있습니다.).

USER 및 USING을 포함하는 피연산자가 없는 CONNECT는 DB2DBDFT 환경 변수를 사용하여 데이터베이스에 응용프로그램 프로세스를 계속 연결할 수 있습니다. 이 방법은 유형 2 CONNECT RESET에 해당되는 방법이나, 사용자 ID와 암호의 사용을 허용합니다.

RESET

사용 가능한 경우, 기본 데이터베이스에 대한 내재적 연결과 같습니다. 기본 데이터베이스를 사용할 수 없으면, 응용프로그램 프로세스의 연결 상태와 해당되는 연결의 상태는 변경되지 않습니다.

기본 데이터베이스의 사용 가능성은 설치 옵션, 환경 변수, 그리고 인증 설정 값에 의해 결정됩니다. 설치 및 환경 변수에 대한 내재적 연결 설정 정보 및 인증 설정 정보에 대해서는 *빠른 시작* 및 *관리 안내서*에서 참조하십시오.

규칙

- 45 페이지의 『분산 작업 단위(DUOW) 의미 정의 옵션』에 설명된 대로, 일련의 연결 옵션에서는 연결 관리의 의미도 다릅니다. 기본값은 처리되는 모든 원시 파일에 지정됩니다. 응용프로그램은 다른 연결 옵션으로 사전 처리 컴파일된 여러 개의 원시 파일로 구성될 수 있습니다.

  SET CLIENT 명령이나 API가 먼저 실행되지 않으면, 런타임 실행된 첫번째 SQL문을 포함하는 원시 파일을 사전에 처리할 때 사용되는 연결 옵션은 유효한 연결 옵션이 됩니다.

  다른 연결 옵션으로 사전 처리된 원시 파일의 CONNECT문이 SET CLIENT 명령이나 API를 끼워 넣지 않고 후속적으로 실행된 경우, 오류(SQLSTATE 08001)가 발생합니다. 일단 SET CLIENT 명령이나 API가 실행되었으면, 응용프로그램의 모든 원시 파일이 사전 처리될 때 사용된 연결 옵션은 무시됩니다.

  624페이지의 예 1에서는 이러한 규칙을 보여줍니다.

- CONNECT TO문을 사용하여 연결을 설정하거나 전환할 수 있어도, 명명된 서버에 대한 현재 또는 휴면 연결이 없을 때만 USER/USING절이 있는 CONNECT TO가 승인됩니다. 연결은 USER/USING절을 갖고 있는 같은 서버에 대한 연결을 발행하기 전에 해제되어야 합니다. 그렇지 않으면, 연결은 거

부됩니다(SQLSTATE 51022). DISCONNECT문 또는 RELEASE문과 그 다음에 COMMIT문을 발행하여 연결을 해제하십시오.

주

- 유형 2의 연결을 갖는 응용프로그램에서 첫번째 SQL문에 대해 내재적 연결이 지원됩니다. 기본 데이터베이스에서 SQL문을 실행하려면, 먼저 연결을 설정하기 위해 CONNECT RESET 또는 CONNECT USER/USING문을 사용해야 합니다. 피연산자가 없는 CONNECT문은 현재 연결이 있는 경우에는 그 연결에 대한 정보를 표시하고, 현재 연결이 없으면, 기본 데이터베이스에 연결하지 않습니다.

유형 1과 유형 2 CONNECT문의 비교:

CONNECT문의 의미는 CONNECT 사전 처리 컴파일러 옵션이나 SET CLIENT API에 의해 판별됩니다(45 페이지의 『분산 작업 단위(DUOW) 의미 정의 옵션』 참조). CONNECT 유형 1 또는 CONNECT 유형 2를 지정할 수 있으며, 프로그램의 CONNECT문들은 각각 유형 1 및 유형 2 CONNECT문으로 알려집니다. 그 의미는 아래에 설명되어 있습니다.

CONNECT TO의 사용:

유형 1	유형 2
각 작업 단위는 하나의 응용프로그램 서버에 대한 연결을 설정할 수 있습니다.	각 작업 단위는 복수의 응용프로그램 서버에 대한 연결만 설정할 수 있습니다.
현재 작업 단위는 다른 응용프로그램 서버에 대한 연결을 허용하기 전에 확약되거나 구간 복원되어야 합니다.	현재 작업 단위는 다른 응용프로그램 서버에 연결하기 전에 확약되거나 구간 복원될 필요가 없습니다.
CONNECT문은 현재 연결을 설정합니다. 후속 SQL 요구는 다른 CONNECT에 의해 변경될 때까지 이 연결에 이송됩니다.	첫번째 연결을 설정중이면, 유형 1 CONNECT와 같습니다. 휴면(dormant) 연결로 전환중이고 SQLRULES가 STD로 설정되어 있으면, SET CONNECTION문이 대신에 사용되어야 합니다.
현재 연결에 연결하는 것은 유효하며 현재 연결은 변경되지 않습니다.	SQLRULES 사전 처리 컴파일러 옵션이 DB2로 설정되면, 유형 1 CONNECT와 같습니다. SQLRULES가 STD로 설정되면, SET CONNECTION문을 대신 사용해야 합니다.

유형 1	유형 2
다른 응용프로그램 서버에 연결하면 현재 연결이 단절됩니다. 새 연결은 현재 연결이 됩니다. 하나의 작업 단위(UOW)에서 하나의 연결만 유지보수됩니다.	다른 응용프로그램 서버에 연결하면 현재 연결이 *휴면(dormant) 상태*가 됩니다. 새 연결은 현재 연결이 됩니다. 복수의 연결이 하나의 작업 단위에서 유지보수될 수 있습니다. CONNECT가 휴면 상태의 연결에 있는 응용프로그램 서버(AS)에 대한 것이면, 그것은 현재 연결이 됩니다. CONNECT를 사용하여 휴면 연결에 연결하는 것은 SQLRULES(DB2)가 지정된 경우에만 허용됩니다. SQLRULES(STD)가 지정되면, SET CONNECTION문이 대신 사용되어야 합니다.
SET CONNECTION문은 유형 1 연결을 지원하나 유일한 유효한 목표는 현재 연결입니다.	SET CONNECTION문은 연결을 휴면에서 현재 상태로 변경하기 위한 유형 2 연결에 대해 지원됩니다.

CONNECT...USER...USING의 사용:

유형 1	유형 2
USER...USING절을 사용하여 연결하면 현재 연결이 끊어지고, 주어진 권한 이름과 암호를 갖는 새로운 연결을 설정합니다.	USER/USING절은 명명된 동일한 서버에 대해 현재 또는 휴면 연결이 전혀 없을 경우에만 허용됩니다.

내재된 CONNECT, CONNECT RESET 및 연결해제의 사용:

유형 1	유형 2
CONNECT RESET은 현재 연결을 끊을 때 사용할 수 있습니다.	CONNECT RESET은 기본 응용프로그램 서버가 시스템에 정의되어 있을 경우 명시적으로 그 서버에 연결하는 것과 같습니다. 연결은 응용프로그램의 정상적인 COMMIT에 의해 끊어질 수 있습니다. 확약 전에, RELEASE문을 사용하여 연결을 해제 보류로 표시하십시오. 그러한 모든 연결은 다음 COMMIT에서 끊어집니다. 또다른 방법은 RELEASE문 대신에 사전 처리 컴파일러 옵션인 DISCONNECT(EXPLICIT), DISCONNECT(CONDITIONAL), DISCONNECT(AUTOMATIC) 또는 DISCONNECT문을 사용하는 것입니다.
CONNECT RESET을 사용하여 현재 연결을 끊은 후, 다음 SQL문이 CONNECT문이 아니면, 기본 응용프로그램 서버가 시스템에 연결되어 있으면 그 서버에 대한 내재적 연결을 수행합니다.	CONNECT RESET은 기본 응용프로그램 서버가 시스템에 정의되어 있는 경우 그 서버에 대한 명시적 연결과 같습니다.
연속 CONNECT RESET을 발행하면 오류가 발생합니다.	SQLRULES(STD)를 지정한 경우에만, 이 옵션은 기존 연결에 대해 CONNECT를 사용하는 것을 허용하지 않으므로 연속 CONNECT RESET을 발행하면 오류가 발생합니다.
CONNECT RESET도 현재 작업 단위를 명시적으로 확약합니다.	CONNECT RESET은 현재 작업 단위를 확약하지 않습니다.
기존 연결이 어떠한 이유로든 시스템에서 끊어지면, 그 다음에 오는 데이터베이스에 대한 비 CONNECT SQL문은 SQLSTATE 08003을 수신하게 됩니다.	기존 연결이 시스템에 의해 끊어지면, COMMIT, ROLLBACK과 SET CONNECTION문은 계속 사용할 수 있습니다.
작업 단위(UOW)는 응용프로그램 프로세스가 성공적으로 종료될 때 명시적으로 확약됩니다.	유형 1과 같습니다.
모든 연결(하나만)은 응용프로그램 프로세스가 종료될 때 끊어집니다.	모든 연결(현재, 휴면 및 해제 보류에 대해 표시된 연결)은 응용프로그램 프로세스가 종료되면 끊어집니다.

CONNECT 실패:

유형 1	유형 2
현재 연결의 유무에 관계없이, CONNECT가 실패하면(지역 디렉토리에 정의되지 않은 서버 이름 이외의 오류로 인해), 응용프로그램 프로세스는 연결되지 않은 상태가 됩니다. 그 다음의 비CONNECT문은 SQLSTATE 08003을 수신합니다.	현재 연결이 있을 때 CONNECT가 실패하면, 현재 연결은 영향을 받지 않습니다. CONNECT 실패시 현재 연결이 전혀 없으면, 프로그램은 연결되지 않은 상태가 됩니다. 그 다음의 비CONNECT문은 SQLSTATE 08003을 수신합니다.

예

예 1: 이 예는 다중 소스 프로그램(상자안에 표시)의 사용을 보여주는 데, 일부는 다중 연결 옵션으로 사전 처리되었으며(코드위에 표시), 이들 중 하나에는 SET CLIENT API 호출(call)이 들어 있습니다.

PGM1: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)

```
...
exec sql CONNECT TO OTTAWA;
exec sql SELECT col1 INTO :hv1
FROM tbl1;
...
```

PGM2: CONNECT(2) SQLRULES(STD) DISCONNECT(AUTOMATIC)

```
...
exec sql CONNECT TO QUEBEC;
exec sql SELECT col1 INTO :hv1
FROM tbl2;
...
```

PGM3: CONNECT(2) SQLRULES(STD) DISCONNECT(EXPLICIT)

```
...
SET CLIENT CONNECT 2  SQLRULES DB2  DISCONNECT EXPLICIT [1]
exec sql CONNECT TO LONDON;
exec sql SELECT col1 INTO
:hv1 FROM tbl3;
...
```

1 주: SET CLIENT API의 실제 구문은 아닙니다.

PGM4: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)

```
   ...
   exec sql CONNECT TO REGINA;
   exec sql SELECT col1 INTO
:hv1 FROM tbl4;
   ...
```

응용프로그램이 PGM1을 실행하고 나서 PGM2를 실행할 경우:

- OTTAWA에 대한 연결이 수행됩니다: connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- QUEBEC에 대한 연결이 SQLSTATE 08001로 실패합니다. SQLRULES와 DISCONNECT는 둘다 다르기 때문입니다.

응용프로그램이 PGM1을 실행하고 나서 PGM3을 실행할 경우:

- OTTAWA에 대한 연결이 수행됩니다: connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- LONDON에 대한 연결이 수행됩니다: connect=2, sqlrules=DB2, disconnect=EXPLICIT

이것은 SET CLIENT API가 두 번째 CONNECT문 전에 수행되므로 성립합니다.

응용프로그램이 PGM1을 실행하고 나서 PGM4를 실행할 경우:

- OTTAWA에 대한 연결이 수행됩니다: connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- REGINA에 대한 연결이 수행됩니다: connect=2, sqlrules=DB2, disconnect=CONDITIONAL

이것은 PGM1에 대한 사전 처리 컴파일러 옵션이 PGM4에 대한 것과 같으므로 성립합니다.

예 2:

이 예는 CONNECT(유형 2), SET CONNECTION, RELEASE, DISCONNECT 문의 상호 관계를 보여줍니다. S0, S1, S2, S3는 네 개의 서버를 나타냅니다.

순서	명령문	현재 서버	휴면 연결	해제 보류
0	명령문 없음	없음	없음	없음
1.	SELECT * FROM TBLA	S0(기본값)	없음	없음
2	CONNECT TO S1 SELECT * FROM TBLB	S1 S1	S0 S0	없음없음
3	CONNECT TO S2 UPDATE TBLC SET ...	S2 S2	S0, S1 S0, S1	없음없음
4	CONNECT TO S3 SELECT * FROM TBLD	S3 S3	S0, S1, S2 S0, S1, S2	없음없음
5	SET CONNECTION S2	S2	S0, S1, S3	없음
6	RELEASE S3	S2	S0, S1	S3
7	COMMIT	S2	S0, S1	없음
8	SELECT * FROM TBLE	S2	S0, S1	없음
9	DISCONNECT S1 SELECT * FROM TBLF	S2 S2	S0 S0	없음없음

CREATE ALIAS

CREATE ALIAS문은 테이블, 뷰, 별명(nickname) 또는 또는 별명(alias)에 대한 별명(alias)을 정의합니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

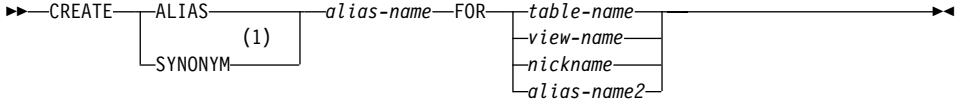
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권에 다음 중 최소한 하나는 포함되어야 합니다.

- SYSADM 또는 DBADM 권한
- 별명의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 별명의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권.

별명으로 언급된 오브젝트를 사용하려면, 오브젝트 그 자체가 사용된 경우에 필요한 특권과 동일한 특권이 그 오브젝트에 필요합니다.

구문



주:

1 CREATE SYNONYM은 다른 SQL implementations CREATE SYNONYM문의 구문 허용 오차를 위한 CREATE ALIAS의 대안으로 받아들여집니다.

설명

별명

별명을 명명합니다. 이 이름은 현재 데이터베이스에 있는 테이블, 뷰, 별명(nickname) 또는 별명(alias)을 식별해서는 안 됩니다.

두 부분으로 이루어진 이름이 지정된 경우, 스키마 이름은 ″SYS″로 시작될 수 없습니다(SQLSTATE 42939).

별명을 정의하는 규칙은 테이블 이름을 정의할 경우에 사용되는 규칙과 같습니다.

FOR *테이블 이름, 뷰 이름, 별명 또는 별명2*

*별명(alias-name)*이 정의되는 테이블, 뷰, 별명(nickname) 또는 별명(alias)을 식별합니다. 다른 별명의 명칭이 제공되면(*별명의 명칭 2*), 정의될(완전 규정화된 양식으로) 새로운 *별명의 명칭*과 같아서는 안 됩니다. *테이블 이름*은 선언된 임시 테이블이 될 수 없습니다(SQLSTATE 42995).

주

- 새로 작성된 별명에 대한 정의는 SYSCAT.TABLES에 저장됩니다.
- 별명은 정의시 존재하지 않는 오브젝트에 대해 정의될 수 있습니다. 오브젝트가 존재하지 않으면 경고가 발행됩니다(SQLSTATE 01522). 그러나, 별명을 포함하는 SQL문이 컴파일될 때 참조된 오브젝트가 있어야 하며, 그 외에는 오류가 발생합니다(SQLSTATE 52004).
- 별명 체인의 다른 별명 부분으로 참조하기 위해 별명을 정의할 수 있으나, 이 체인은 SQL문에서 사용될 때 단일 별명과 같은 제한사항을 갖습니다. 별명 체인은 단일 별명과 같은 방법으로 해결됩니다. 뷰 정의, 패키지의 명령문 또는 트리거에 사용된 별명이 별명 체인에 포인터를 지정하면, 체인의 각 별명에서의 뷰, 패키지 또는 트리거에 대한 종속성이 기록됩니다. 별명 체인의 반복 주기는 허용되지 않으며 별명 정의시 감지됩니다.
- 아직 없는 스키마 이름을 사용하여 별명을 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.

예

예 1: HEDGES는 테이블 T1에 대해 별명을 작성하려 합니다(둘다 규정화되지 않았음).

```
CREATE ALIAS A1 FOR T1
```

별명 HEDGES.A1이 HEDGES.T1에 대해 작성됩니다.

예 2: HEDGES는 테이블에 대해 별명을 작성하려 합니다(둘다 규정화되었음).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
```

별명 HEDGES.A1이 MCKNIGHT.T1에 대해 작성됩니다.

예 3: HEDGES는 테이블에 대해 별명을 작성하려 합니다. (서로 다른 스키마의 별명. HEDGES는 DBADM이 아니며, HEDGES는 스키마 MCKNIGHT에 대해 CREATEIN을 가지고 있지 않습니다.)

```
CREATE ALIAS MCKNIGHT.A1 FOR MCKNIGHT.T1
```

이 예는 실패합니다(SQLSTATE 42501).

예 4: HEDGES는 정의되지 않은 테이블에 대해 별명을 작성하려 합니다(둘다 규정화되었음. FUZZY.WUZZY는 존재하지 않습니다.).

```
CREATE ALIAS HEDGES.A1 FOR FUZZY.WUZZY
```

이 명령문은 성공하나 경고가 발행됩니다(SQLSTATE 01522).

예 5: HEDGES는 별명에 대해 별명을 작성하려 합니다(둘다 규정화되었음).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
CREATE ALIAS HEDGES.A2 FOR HEDGES.A1
```

첫번째 명령문은 성공합니다(예 2).

두 번째 명령문은 성공하고 MCKNIGHT.T1을 참조하는 HEDGES.A1을 참조하는 HEDGES.A2로 구성되는 별명 체인이 작성됩니다. HEDGES가 MCKNIGHT.T1에 대한 임의의 특권을 갖고 있는 지의 여부는 문제가 되지 않습니다. 테이블 특권에 관계없이 별명이 작성됩니다.

예 6: A1을 별명(nickname) FUZZYBEAR에 대한 별명(alias)으로서 지정하십시오.

```
CREATE ALIAS A1 FOR FUZZYBEAR
```

예 7: 커다란 조직에 D108로 번호 매긴 재정 부서와 D577이라 번호매긴 인사 부서가 있습니다. D108은 DB2 RDBMS에 상주하는 테이블에 특정 정보를 보관합니다. D577은 Oracle RDBMS에 상주하는 테이블에 특정 레코드를 보관합니다. DBA는 연합 시스템 내에서 데이터 소스로서 두 개의 RDBMS를 정의하며, 테이블에 각각 DEPTD108과 DEPTD577의 별명(nickname)을 부여합니다. 연합 시스템 사용자는 이들 테이블간에 조인을 작성해야 하지만, 영숫자 별명보다 더 의미있는 이름으로 이들을 참조하고자 할 것입니다. 따라서 사용자는 DEPTD108에 대한 별명으로 FINANCE를 DEPTD577에 대한 별명으로 PERSONNEL을 정의합니다.

```
CREATE ALIAS FINANCE FOR DEPTD108
CREATE ALIAS PERSONNEL FOR DEPTD577
```

CREATE BUFFERPOOL

CREATE BUFFERPOOL문은 데이터베이스 관리 프로그램이 사용할 새로운 버퍼 풀을 생성합니다. 버퍼 풀 정의가 트랜잭션 가능하고 확약시 항목들이 카탈로그 테이블에 반영된다고 하여도, 다음 번에 데이터베이스가 시작될 때까지 버퍼 풀은 활성화되지 않습니다.

특정 파티션 또는 노드에서 *크기*를 대체할 능력이 있는 경우, 파티션된 데이터베이스에서는 각 파티션 또는 노드에 대한 기본 버퍼 풀 정의가 지정됩니다. 또한 파티션된 데이터베이스에서, 노드 그룹이 지정되어 있지 않은 경우 모든 파티션에 버퍼 풀이 정의됩니다. 노드 그룹이 지정되는 경우, 이러한 노드 그룹에 있는 파티션에만 버퍼 풀이 작성됩니다.

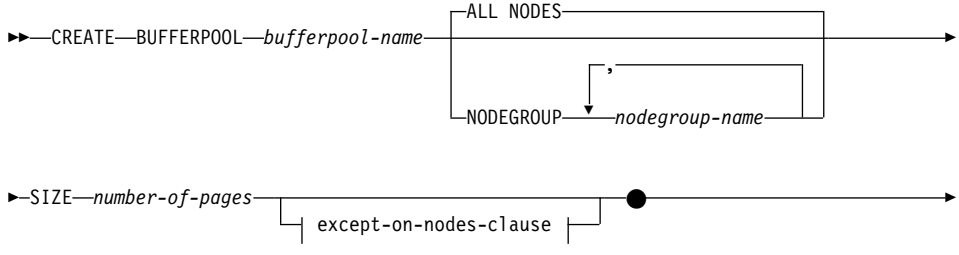
호출

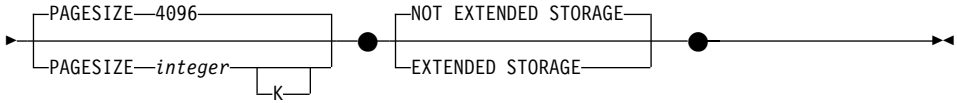
이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

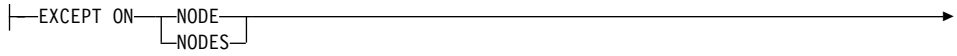
명령문의 권한 부여 ID는 SYSCTRL 또는 SYSADM 권한이 있어야 합니다.

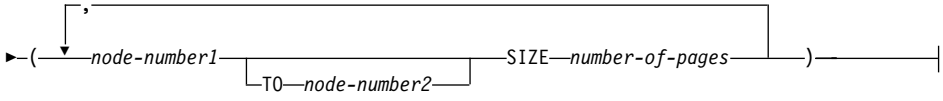
구문





except-on-nodes-clause:





설명

버퍼풀 이름

버퍼 풀에 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. 이것은 SQL 식별자(일반 또는 분리 식별자)입니다. *버퍼 풀 이름*은 카탈로그에 이미 존재하는 버퍼 풀을 나타내면 안 됩니다(SQLSTATE 42710). *버퍼 풀 이름*은 "SYS" 또는 "IBM"으로 시작하면 안 됩니다(SQLSTATE 42939).

ALL NODES

데이터베이스의 모든 파티션에 이 버퍼 풀이 작성됩니다.

NODEGROUP *노드 그룹 이름, ...*

버퍼 풀 정의를 적용할 수 있는 노드 그룹(들)을 식별합니다. 지정될 경우, 이러한 노드 그룹에 있는 파티션에만 이 버퍼 풀이 생성됩니다. 각 노드 그룹은 데이터베이스에 존재하고 있어야 합니다(SQLSTATE 42704). NODEGROUP 키워드가 지정되지 않은 경우, 이 버퍼 풀은 모든 파티션(및 데이터베이스에 뒤이어 추가되는 모든 파티션)에 작성됩니다.

SIZE *페이지 수*

페이지 수로 지정되는 버퍼 풀의 *크기*. [69] 파티션된 데이터베이스에서, 이 *크기*는 버퍼 풀이 있는 모든 파티션의 *기본 크기*가 됩니다.

---

69. (-1)의 값으로 *크기*를 지정할 수 있는 데, 이 값은 BUFFPAGE 데이터베이스 구성 매개변수의 버퍼 풀 *크기*를 취해야 함을 나타냅니다.

except-on-nodes절

버퍼 풀의 *크기*가 기본 *크기*와 다른 파티션 또는 파티션들을 지정합니다. 이 절이 지정되지 않은 경우, 모든 파티션이 이 버퍼 풀에 지정된 것과 같은 *크기*의 버퍼 풀을 갖게 됩니다.

EXCEPT ON NODES

지정된 특정 파티션을 가리키는 키워드. NODE는 NODES와 동의어입니다.

노드 번호1

버퍼 풀이 작성된 파티션에 포함되어 있는 특정 파티션 번호를 지정합니다.

TO *노드 번호2*

파티션 번호의 범위를 지정합니다. *노드 번호2*의 값은 *노드 번호1*의 값보다 *크거나* 같아야 합니다(SQLSTATE 428A9). 지정된 파티션 번호 사이의 모든 파티션은 버퍼 풀이 작성된 파티션에 포함되어 있어야 합니다(SQLSTATE 42729).

SIZE *페이지 수*

페이지 수로 지정되는 버퍼 풀의 *크기*.

PAGESIZE *정수* **[K]**

버퍼 풀에 사용되는 페이지 *크기*를 정의합니다. *정수*에 대한 유효값은 접미부 K 없이 4 096, 8 192, 16 384 또는 32 768입니다. *정수*에 대한 유효값은 접미부 K 없이 4, 8, 16 또는 32입니다. 페이지 *크기*가 이러한 값 중의 하나가 아니면 오류가 발생합니다(SQLSTATE 428DE). 기본값은 4 096바이트(4K) 페이지입니다. *정수*와 K 사이에는 공백이 없을 수도 있고 임의의 수의 공백이 들어갈 수도 있습니다.

EXTENDED STORAGE

확장 저장영역 구성이 on인 경우, [70] 이 버퍼 풀 외부로부터 이주되고 있는 페이지는 확장 저장영역에 캐쉬됩니다.

---

70. 확장 저장영역 구성은 데이터베이스 구성 매개변수 NUM_ESTORE_SEGS 및 ESTORE_SEG_SIZE를 0이 아닌 값으로 설정함으로써 작동됩니다. *관리 안내서*에서 자세한 내용을 참조하십시오.

NOT EXTENDED STORAGE

확장 저장영역 구성이 작동되고 있는 경우에도, 이 버퍼 풀로부터 이주되고 있는 페이지는 확장 저장영역에 캐쉬되지 않습니다.

주

- 다음 번에 데이터베이스가 시작될 때까지, 작성된 모든 테이블 공간은 동일한 테이블 공간을 가진 이미 사용중인 버퍼 풀을 사용하게 됩니다. 데이터베이스는 영향을 미치는 새 버퍼 풀에 대한 테이블 공간 할당에 대해 재시작 되어야 합니다.
- 데이터베이스 관리 프로그램과 응용프로그램의 나머지 부분에 대해서뿐 아니라 전체 버퍼 풀에 대해 머신상에 충분한 실제 메모리가 있어야 합니다. 전체 버퍼 풀에 대한 메모리를 확보할 수 없는 경우, DB2는 기본 버퍼 풀만을 시작하려 합니다. 이것이 성공적이지 않으면 DB2는 최소한의 기본 버퍼 풀을 시동합니다. 이러한 경우, 사용자에게 경고가 리턴되며(SQLSTATE 01626) 모든 테이블 공간의 페이지는 기본 버퍼 풀을 사용하게 됩니다.

CREATE DISTINCT TYPE

CREATE DISTINCT TYPE문은 구별 유형을 정의합니다. 구별 유형은 내장 데이터 유형 중 하나를 기반으로 합니다. 명령문이 성공적으로 실행되면 구별 유형과 이 소스 유형 사이에서 변환하기 위한 함수가 생성되고, 선택적으로 구별 유형에 사용하도록 비교 연산자(=, <>, <, <=, > 및 >=)가 생성됩니다.

호출

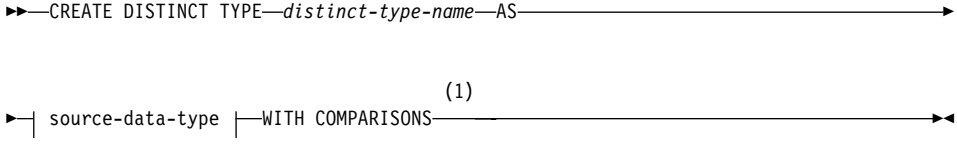
이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

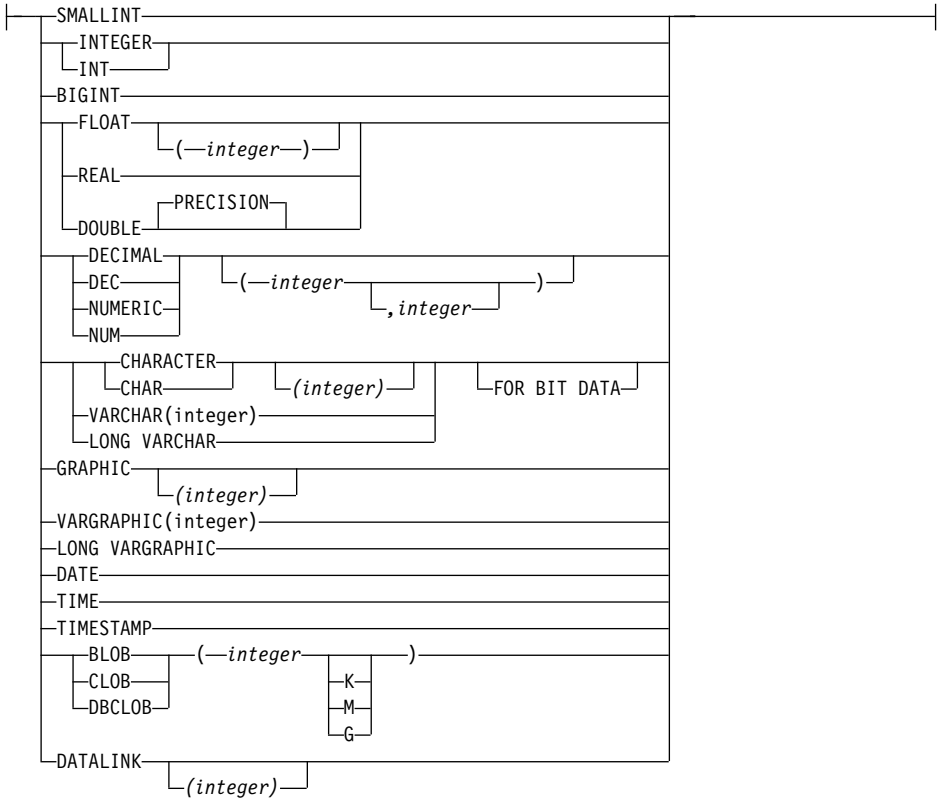
명령문의 권한 부여 ID에서 갖고 있는 특권에 다음 중 최소한 하나는 포함되어야 합니다.

- SYSADM 또는 DBADM 권한
- 구별 유형의 스키마 이름이 기존의 스키마를 가리키지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 구별 유형의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

구문



source-data-type:



주:

1 지원되지 않는 LOB, LONG VARCHAR, LONG VARGRAPHIC 및 DATALINK를 제외한 모든 소스 데이터 유형에 대해 필요합니다.

설명

구별 유형 이름(distinct-type-name)

구별 유형을 명명합니다. 암시적 및 명시적 규정자를 포함하는 이 이름은 카탈로그에 설명된 구별 유형을 식별하지 않아야 합니다. 규정되지 않은 이름은 소스 데이터 유형의 이름이나 BOOLEAN과 같으면 안 됩니다(SQLSTATE 42918).

동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER

precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다.

스키마 이름(내재적 또는 명시적)은 8 바이트를 초과할 수 없습니다 (SQLSTATE 42622).

술어에서 키워드로 사용되는 이름의 일부는 시스템에서 사용할 수 있도록 예약되어 있어서, *구별 유형 이름*으로 사용할 수 없습니다. SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 등이 이러한 이름이며, 217 페이지의 『기본 술어』에 기술된 비교 연산자도 여기에 포함됩니다. 이 규칙을 지키지 못하면 오류가 발생됩니다(SQLSTATE 42939).

두 부분의 *구별 유형 이름*을 지정하면, 스키마 이름은 "SYS"로 시작할 수 없습니다. 그렇지 않으면, 오류(SQLSTATE 42939)가 발생합니다.

소스 데이터 유형

구별 유형의 내부 표시에 대한 기준으로 사용되는 데이터 유형을 지정합니다. 구별 유형과 다른 데이터 유형과의 연관성에 대해서는 101 페이지의 『구별 유형』에서 참조하고, 데이터 유형에 대해서는 800 페이지의 『CREATE TABLE』에서 참조하십시오.

WITH COMPARISONS

구별 유형의 두 인스턴스를 비교하기 위해 시스템에서 생성한 비교 연산자가 작성되도록 지정합니다. 이러한 키워드는 소스 데이터 유형이 BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC 또는 DATALINK인 경우 지정해서는 안되며, 지정하면 경고가 리턴되고(SQLSTATE 01596) 비교 연산자가 생성되지 않습니다. 모든 다른 소스 데이터 유형의 경우, WITH COMPARISONS 키워드는 필수입니다.

주

- 아직 없는 스키마 이름을 사용하여 구별 유형을 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 해당 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.

- 소스 유형 사이의 변환을 위해 다음의 함수가 생성됩니다.
  - 구별 유형에서 소스 유형으로 변환시키기 위한 하나의 함수
  - 소스 유형에서 구별 유형으로 변환시키기 위한 하나의 함수
  - 소스 유형이 SMALLINT인 경우 INTEGER에서 구별 유형으로 변환시키기 위한 하나의 함수
  - 소스 유형이 CHAR인 경우 VARCHAR에서 구별 유형으로 변환시키기 위한 하나의 함수
  - 소스 유형이 GRAPHIC인 경우 VARGRAPHIC에서 구별 유형으로 변환시키기 위한 하나의 함수

일반적으로 이 함수들은 다음 형식을 갖습니다.

```
CREATE FUNCTION source-type-name (distinct-type-name)
   RETURNS source-type-name ...
CREATE FUNCTION distinct-type-name (source-type-name)
   RETURNS distinct-type-name ...
```

소스 유형이 매개변수 유형인 경우, 구별 유형에서 소스 유형으로 변환시키는 함수는 매개변수 없이 소스 유형 이름을 함수 이름으로 가지게 됩니다(639 페이지의 표20에서 자세한 내용을 참조하십시오.). 소스의 리턴 값 유형에는 CREATE DISTINCT TYPE문에서 제공되는 매개변수가 포함됩니다. 소스 유형을 구별 유형으로 변환하는 함수는 소스 유형이면서 매개변수를 포함하는 입력 매개변수를 갖습니다. 예:

```
   CREATE DISTINCT TYPE T_SHOESIZE AS CHAR(2)
WITH COMPARISONS
   CREATE DISTINCT TYPE T_MILES AS DOUBLE
WITH COMPARISONS
```

다음 함수를 생성합니다.

```
   FUNCTION CHAR (T_SHOESIZE) RETURNS CHAR (2)
   FUNCTION T_SHOESIZE (CHAR (2))
RETURNS T_SHOESIZE
   FUNCTION DOUBLE (T_MILES) RETURNS DOUBLE
   FUNCTION T_MILES (DOUBLE) RETURNS T_MILES
```

생성된 변환 함수의 스키마는 구별 유형의 스키마와 같습니다. 이 이름과, 동일한 시그니처를 갖고 있는 다른 어떤 함수도 데이터베이스에 존재하지 않을 것입니다 (SQLSTATE 42710).

다음 표는 사전 정의된 모든 데이터 유형에 대해 구별 유형에서 소스 유형으로, 그리고 소스 유형에서 구별 유형으로 변환하기 위한 함수들의 이름을 제공합니다.

표 20. 구별 유형에서의 변환(CAST) 함수

소스 유형 이름	함수 이름	매개변수	리턴 유형
CHAR	<distinct>	CHAR (n)	<distinct>
	CHAR	<distinct>	CHAR (n)
	<distinct>	VARCHAR (n)	<distinct>
VARCHAR	<distinct>	VARCHAR (n)	<distinct>
	VARCHAR	<distinct>	VARCHAR (n)
LONG VARCHAR	<distinct>	LONG VARCHAR	<distinct>
	LONG_VARCHAR	<distinct>	LONG VARCHAR
CLOB	<distinct>	CLOB (n)	<distinct>
	CLOB	<distinct>	CLOB (n)
BLOB	<distinct>	BLOB (n)	<distinct>
	BLOB	<distinct>	BLOB (n)
GRAPHIC	<distinct>	GRAPHIC (n)	<distinct>
	GRAPHIC	<distinct>	GRAPHIC (n)
	<distinct>	VARGRAPHIC (n)	<distinct>
VARGRAPHIC	<distinct>	VARGRAPHIC (n)	<distinct>
	VARGRAPHIC	<distinct>	VARGRAPHIC (n)
LONG VARGRAPHIC	<distinct>	LONG VARGRAPHIC	<distinct>
	LONG_VARGRAPHIC	<distinct>	LONG VARGRAPHIC
DBCLOB	<distinct>	DBCLOB (n)	<distinct>
	DBCLOB	<distinct>	DBCLOB (n)
SMALLINT	<distinct>	SMALLINT	<distinct>
	<distinct>	INTEGER	<distinct>
	SMALLINT	<distinct>	SMALLINT
INTEGER	<distinct>	INTEGER	<distinct>
	INTEGER	<distinct>	INTEGER

표 20. 구별 유형에서의 변환(CAST) 함수 (계속)

소스 유형 이름	함수 이름	매개변수	리턴 유형
BIGINT	<distinct>	BIGINT	<distinct>
	BIGINT	<distinct>	BIGINT
DECIMAL	<distinct>	DECIMAL (p,s)	<distinct>
	DECIMAL	<distinct>	DECIMAL (p,s)
NUMERIC	<distinct>	DECIMAL (p,s)	<distinct>
	DECIMAL	<distinct>	DECIMAL (p,s)
REAL	<distinct>	REAL	<distinct>
	<distinct>	DOUBLE	<distinct>
	REAL	<distinct>	REAL
FLOAT(*n*) 여기서 *n*<=24	<distinct>	REAL	<distinct>
	<distinct>	DOUBLE	<distinct>
	REAL	<distinct>	REAL
FLOAT(*n*) 여기서 *n*>24	<distinct>	DOUBLE	<distinct>
	DOUBLE	<distinct>	DOUBLE
FLOAT	<distinct>	DOUBLE	<distinct>
	DOUBLE	<distinct>	DOUBLE
DOUBLE	<distinct>	DOUBLE	<distinct>
	DOUBLE	<distinct>	DOUBLE
DOUBLE PRECISION	<distinct>	DOUBLE	<distinct>
	DOUBLE	<distinct>	DOUBLE
DATE	<distinct>	DATE	<distinct>
	DATE	<distinct>	DATE
TIME	<distinct>	TIME	<distinct>
	TIME	<distinct>	TIME
TIMESTAMP	<distinct>	TIMESTAMP	<distinct>
	TIMESTAMP	<distinct>	TIMESTAMP
DATALINK	<distinct>	DATALINK	<distinct>
	DATALINK	<distinct>	DATALINK

주: NUMERIC과와 FLOAT는 이식 가능한 응용프로그램에 대해 사용자 정의 유형을 작성할 때 사용하지 않는 것이 좋습니다. DECIMAL 및 DOUBLE을 대신 사용해야 합니다.

위의 표에 설명된 함수는 구별 유형이 정의될 때 자동으로 생성되는 유일한 함수입니다. 결과적으로, 구별 유형으로 사용자 정의 함수를 등록하는 데 CREATE FUNCTION문(653 페이지의 『CREATE FUNCTION』 참조)을 사용할 때까지는 내장 함수(AVG, MAX, LENGTH 등)가 구별 유형에서 전혀 지원되지 않습니다. 여기서, 그 사용자 정의 함수는 해당 내장 함수를 기준으로 합니다. 특히, 내장 컬럼 함수를 근거로 하는 사용자 정의 함수를 등록할 수 있습니다.

WITH COMPARISONS절을 사용하여 구별 유형이 작성될 때, 시스템 생성 비교 연산자가 제공됩니다. 이러한 비교 연산자를 작성하면, SYSCAT.FUNCTIONS 카탈로그 뷰에 새 함수에 대한 항목이 생성됩니다.

구별 유형의 스키마 이름은 이러한 연산자의 성공적인 사용과 SQL문에서의 유형 변환(cast) 함수의 경우 SQL 경로에 포함되어야 합니다(1174 페이지의 『SET PATH』, 또는 *응용프로그램 개발 안내서*에 설명된 FUNCPATH BIND 참조).

예

예 1: INTEGER 데이터 유형에 기초한 구별 유형 SHOESIZE를 작성합니다.

```
CREATE DISTINCT TYPE SHOESIZE AS INTEGER WITH COMPARISONS
```

이는 또한, 비교 연산자(=, <>, <, <=, >, >=)와 두 개의 유형변환(cast) 함수 INTEGER(SHOESIZE)(INTEGER를 리턴함)와 SHOESIZE(INTEGER)(SHOESIZE를 리턴함)도 생성합니다.

예 2: DOUBLE 데이터 유형에 기초한 구별 유형 MILES를 생성합니다.

```
CREATE DISTINCT TYPE MILES AS DOUBLE WITH COMPARISONS
```

이는 또한, 비교 연산자(=, <>, <, =, >, >=)와 두 개의 유형변환(cast) 함수 DOUBLE(MILES)(DOUBLE을 리턴함)와 MILES(DOUBLE)(MILES를 리턴함)도 생성합니다.

CREATE EVENT MONITOR

CREATE EVENT MONITOR문은 데이터베이스를 사용할 때 발생하는 특정 이벤트를 기록할 모니터를 정의합니다. 각 이벤트 모니터의 정의도 데이터베이스에서 이벤트를 기록해야 하는 위치를 지정합니다.

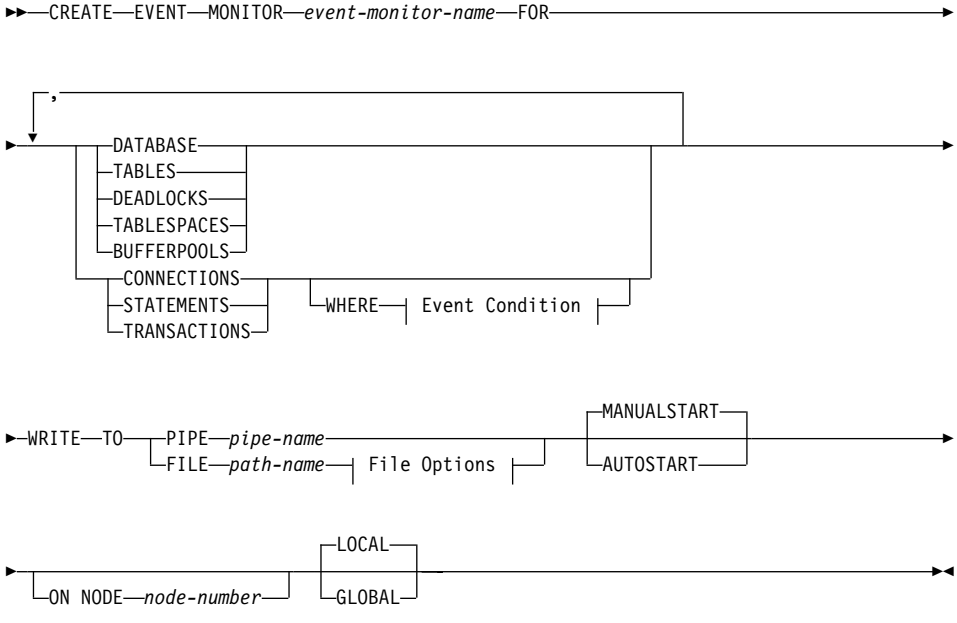
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

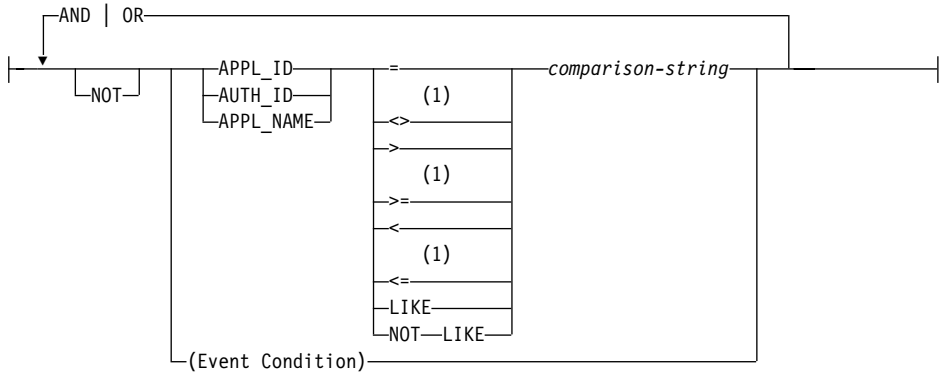
권한 부여

권한 부여 ID에서 갖고 있는 특권에는 SYSADM 또는 DBADM 권한을 포함해야 합니다(SQLSTATE 42502).

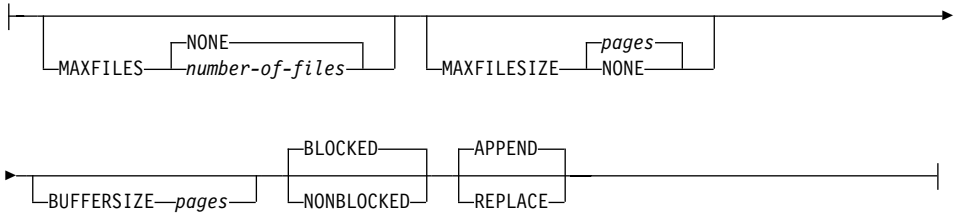
구문



Event Condition:



File Options:



주:

1 다른 형태의 연산자도 지원됩니다. 217 페이지의 『기본 술어』에서 자세한 내용을 참조하십시오.

설명

이벤트 모니터 이름

이벤트 모니터를 명명합니다. 이 이름은 한 부분의 이름입니다. 이것은 SQL 식별자(일반 또는 분리 식별자)입니다. *이벤트 모니터 이름*은 카탈로그에 이미 존재하는 이벤트 모니터와 구분되어야 합니다(SQLSTATE 42710).

FOR

기록할 레코드의 유형을 소개합니다.

DATABASE

마지막 응용프로그램이 데이터베이스에서 연결 해제될 때 이벤트 모니터가 데이터베이스 이벤트를 기록합니다.

TABLES

마지막 응용프로그램이 데이터베이스에서 연결 해제될 때 사용중인 각 테이블에 대해 이벤트 모니터가 테이블 이벤트를 기록합니다. 사용 중인 테이블은 데이터베이스에 처음 연결한 이후 변경된 테이블입니다.

DEADLOCKS

교착 상태가 발생할 때마다 이벤트 모니터가 교착 상태를 기록합니다.

TABLESPACES

마지막 응용프로그램이 데이터베이스에서 연결 해제될 때 각 테이블 공간에 대해 이벤트 모니터가 테이블 공간 이벤트를 기록합니다.

BUFFERPOOLS

마지막 응용프로그램이 데이터베이스에서 연결 해제될 때 이벤트 모니터가 버퍼 풀 이벤트를 기록합니다.

CONNECTIONS

마지막 응용프로그램이 데이터베이스에서 연결 해제될 때 이벤트 모니터가 연결 이벤트를 기록합니다.

STATEMENTS

SQL문이 실행을 종료할 때마다 이벤트 모니터가 명령문을 기록합니다.

TRANSACTIONS

트랜잭션이 종료될 때마다(즉, 확약 또는 구간 복원 조작이 있을 때마다) 이벤트 모니터가 트랜잭션 이벤트를 기록합니다.

WHERE *이벤트 조건*

CONNECTION, STATEMENT 또는 TRANSACTION 이벤트를 발생시키는 연결을 판별하는 필터를 정의합니다. 이벤트 조건의 결과가 특정 연결에 대해 참이면, 그 연결은 요구된 이벤트를 생성합니다.

이 절은 표준 검색 조건과 혼돈하지 말아야 하는 WHERE절의 특수 양식입니다.

응용프로그램이 특정 이벤트 모니터에 대한 이벤트를 생성하는지 판별하기 위해 WHERE 절이 평가됩니다.

1. 이벤트 모니터가 처음 작동될 때 각 활동중인 연결에 대해.
2. 연결시 데이터베이스에 대한 새로운 각 연결에 대해 후속적으로.

WHERE절은 각 이벤트에 대해 평가되지 않습니다.

WHERE절을 지정하지 않으면 지정된 이벤트 유형의 모든 이벤트가 모니터됩니다.

APPL_ID

연결에서 CONNECTION, STATEMENT 또는 TRANSACTION 이벤트(지정된 것)를 생성해야 하는 지를 판별하기 위해 각 연결의 응용프로그램 ID가 *비교 문자열*과 비교되어야 하는 지를 지정합니다.

AUTH_ID

연결에서 CONNECTION, STATEMENT 또는 TRANSACTION 이벤트(지정된 것)를 생성해야 하는 지를 판별하기 위해 각 연결의 권한 부여 ID가 *비교 문자열*과 비교되어야 하는 지를 지정합니다.

APPL_NAME

연결에서 CONNECTION, STATEMENT 또는 TRANSACTION 이벤트(지정된 것)를 생성해야 하는 지를 판별하기 위해 각 연결의 응용프로그램 이름이 *비교 문자열*과 비교되어야 하는 지를 지정합니다.

응용프로그램 이름은 마지막 경로 분리문자 다음에 오는 응용프로그램 파일 이름의 처음 20바이트입니다.

비교 문자열

데이터베이스에 연결된 각 응용프로그램의 APPL_ID, AUTH_ID나 APPL_NAME으로 비교되는 문자열. *comparison-string*은 문자열 상수여야 합니다(즉, 호스트 변수와 다른 문자열 표현식은 허용되지 않습니다.).

WRITE TO

데이터에 대한 목표를 소개합니다.

PIPE

이벤트 모니터 데이터의 목표가 named pipe 파이프임을 지정합니다. 이

벤트 모니터는 데이터를 단일 스트림의 파이프에 기록합니다(즉, 그것이 단일이면, 무한정으로 긴 파일임). 데이터를 파이프에 기록할 때, 이벤트 모니터는 블록화된 기록사항을 수행하지 않습니다. 파이프 버퍼에 공간이 없으면, 이벤트 모니터에서는 데이터를 버립니다. 데이터 유실이 없도록 하려면, 즉시 데이터를 읽도록 해야 합니다.

파이프 이름

이벤트 모니터가 데이터를 기록할 파이프의 이름(AIX에서 FIFO).

파이프의 명명 규칙은 플랫폼에 따라 다릅니다. UNIX 운영 시스템에서, 파이프 이름은 파일 이름과 같이 취급됩니다. 그러므로, 상대 파이프 이름이 허용되고, 상대 경로 이름과 같이 취급됩니다 (아래의 *경로 이름* 참조). 그러나, OS/2, Windows 95 및 Windows NT에는 파이프 이름에 대한 특수한 구문이 있습니다. 그러므로, OS/2, Windows 95 및 Windows NT의 경우 절대 파이프 이름이 필요합니다.

파이프의 존재여부는 이벤트 모니터 작성시 점검되지 않습니다. 이벤트 모니터가 활성화될 때 읽을 파이프를 작성하고 여는 것은 모니터링 응용프로그램에서 수행됩니다. 이 때 파이프를 읽을 수 없으면, 이벤트 모니터 자체가 작동중지되고 오류가 기록됩니다(즉, 이벤트 모니터가 데이터베이스 시작시 AUTOSTART 옵션에 의해 활성화된 경우, 이벤트 모니터는 시스템 오류 로그에 오류를 기록합니다.). 이벤트 모니터가 SET EVENT MONITOR STATE SQL문으로 활성화된 경우, 그 명령문은 실패합니다(SQLSTATE 58030).

FILE

이벤트 모니터 데이터의 목표가 파일(또는 파일 세트)임을 나타냅니다. 이벤트 모니터는 데이터 스트림을 확장자 “evt”를 갖는 일련의 8문자 순서화 파일로 작성합니다(예를 들면, 00000000.evt, 00000001.evt 및 00000002.evt). 데이터는 더 작은 여러 조각으로 분리되더라도 하나의 논리 파일로 간주됩니다(즉, 데이터 스트림의 시작은 00000000.evt 파일의 첫번째 바이트이고, 데이터 스트림의 끝은 nnnnnnnn.evt 파일의 마지막 바이트임).

각 파일의 최대 *크기*는 파일의 최대 수로도 정의될 수 있습니다. 이벤트 모니터는 단일 이벤트 레코드를 두 파일에 걸쳐 분리시킬 수 없습

니다. 그러나, 이벤트 모니터는 관련 레코드들을 두 개의 다른 파일로 작성할 수 있습니다. 이벤트 파일을 처리할 때 그러한 관련 정보를 추적하는 것은 이 데이터를 사용하는 응용프로그램에서 수행됩니다.

경로 이름

이벤트 모니터가 이벤트 파일 데이터를 기록해야 하는 디렉토리의 이름. 경로는 서버에 알려져 있어야 하나, 경로 그 자체는 다른 파티션이나 노드에 상주할 수 있습니다(예를 들어, UNIX용 시스템의 경우는 NFS 마운트 파일이 될 수 있습니다.). *경로 이름*을 지정할 때 리터럴을 사용해야 합니다.

디렉토리는 CREATE EVENT MONITOR 수행시 존재해서는 안됩니다. 그러나, 이벤트 모니터가 활성화될 때 목표 경로의 존재 여부에 대해 점검합니다. 이 때, 목표 경로가 존재하지 않으면 오류(SQLSTATE 428A3)가 발생합니다.

절대 경로(AIX상에서는 루트 디렉토리로 시작하고 OS/2, Windows 95 및 Windows NT에서는 디스크 식별자로 시작하는 경로)가 지정되는 경우, 그 경로가 사용되는 경로입니다. 상대 경로(루트로 시작하지 않는 경로)를 지정할 경우, 데이터베이스 디렉토리에 있는 DB2EVENT 디렉토리에 관련되는 경로가 사용됩니다.

상대 경로를 지정하면, DB2EVENT 디렉토리를 사용하여 그 경로를 절대 경로로 변환합니다. 그러면, 절대 및 상대 경로 사이에 어떤 구별도 만들어지지 않습니다. 절대 경로는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 저장됩니다.

같은 목표 경로를 갖는 두 개 이상의 이벤트 모니터를 지정할 수 있습니다. 그러나, 처음에 이벤트 모니터 중 하나가 활성화되고 대상 디렉토리가 비어 있지 않으면, 다른 이벤트 모니터 중 어떤 것도 활성화시킬 수 없습니다.

파일 옵션

파일 형식에 대한 옵션을 지정합니다.

MAXFILES NONE

이벤트 모니터가 작성하는 이벤트 파일 수로 제한되지 않음을 지정합니다. 이것은 기본값입니다.

MAXFILES *파일 수*

언제든지 특정 이벤트 모니터에 대해 존재할 이벤트 모니터 파일 수에 제한이 있음을 지정합니다. 이벤트 모니터가 다른 파일을 작성해야 할 때마다, 디렉토리에 있는 .evt 파일의 수가 *파일의 수*보다 반드시 적도록 점검해야 합니다. 이미 이 한계에 도달해 있으면, 이벤트 모니터는 스스로 작동중지됩니다.

응용프로그램이 이벤트 파일이 작성된 후에 디렉토리에서 파일을 제거할 경우, 이벤트 모니터가 작성할 수 있는 총 파일 수는 *파일의 수*를 초과할 수 있습니다. 이 옵션은 이벤트 데이터가 지정된 디스크 공간량보다 많이 사용되지 않도록 사용자가 보증할 수 있도록 하기 위해 제공됩니다.

MAXFILESIZE *페이지*

각 이벤트 모니터 파일의 *크기*에 제한이 있음을 지정합니다. 이벤트 모니터가 새 이벤트 레코드를 파일에 기록할 때마다, 모니터는 파일이 *페이지*(4K 페이지 수 단위)보다 커지지 않도록 점검합니다. 결과 파일이 너무 크면, 이벤트 모니터는 다음 파일로 전환합니다. 이 옵션의 기본값은 다음과 같습니다.

- OS/2, Windows 95 및 Windows NT - 200 4K 페이지
- UNIX - 1000 4K 페이지

페이지 수는 최소한 이벤트 버퍼의 *크기*(페이지 단위)보다 많아야 합니다. 이 요건이 만족되지 않으면, 오류(SQLSTATE 428A4)가 발생합니다.

MAXFILESIZE NONE

파일 *크기*에 제한이 없음을 지정합니다. MAXFILESIZE NONE이 지정된 경우, MAXFILES 1도 지정해야 합니다. 이 옵션은 한 파일에 특수 이벤트 모니터에 대한 모든 이벤트 데이터를 포함함을 의미합니다. 이 경우 이벤트 파일은 00000000.evt입니다.

BUFFERSIZE *페이지*

이벤트 모니터 버퍼의 *크기*를 지정합니다(4K 페이지 수 단위). 모든 이벤트 모니터 파일 I/O는 이벤트 모니터의 성능이 향상되도록 버퍼화됩니다. 버퍼가 크면 클수록, 이벤트 모니터에 의해 수행되는 I/O는 적어집니다. 활발하게 활동중인 이벤트 모니터는 상대적으로 활동이 적은 이벤트 모니터보다 더 큰 버퍼를 갖고 있어야 합니다. 모니터가 시작되면, 지정된 *크기*의 두 버퍼가 할당됩니다. 이벤트 모니터는 비동기 I/O를 허용하기 위해 두배 버퍼링을 사용합니다.

각 버퍼의 최소 및 기본 *크기*는(이 옵션을 지정하지 않을 경우) 4 페이지(즉, 두 개의 버퍼, *크기*는 각각 16 K)입니다. 버퍼의 최대 *크기*는 버퍼가 힙으로부터 할당되므로 모니터 힙(MON_HEAP)의 *크기*로 제한됩니다. 동시에 많은 이벤트 모니터를 사용할 경우, MON_HEAP 데이터베이스 구성 매개변수를 증가시키십시오.

해당 데이터를 파이프에 기록하는 이벤트 모니터도 *크기*로 각각 두 개의 내부(구성 가능하지 않은) 버퍼를 갖고 있습니다. 이 버퍼들은 또한 모니터 힙(MON_HEAP)으로부터 할당됩니다. 파이프 목표를 갖는 각 활동 이벤트 모니터에 대해, 데이터베이스 힙(heap)의 *크기*를 2페이지씩 증가시키십시오.

BLOCKED

이벤트를 생성하는 각 에이전트가 두 이벤트 버퍼 모두가 가득찼는지를 판별할 경우에 이벤트 버퍼가 디스크에 기록되는 동안 기다려야 함을 지정합니다. 이벤트 데이터가 유실되지 않도록 BLOCKED를 선택해야 합니다. 기본 옵션입니다.

NONBLOCKED

이벤트를 생성하는 각 에이전트가 두 이벤트 버퍼 모두가 가득찼는지를 판별할 경우에 이벤트 버퍼가 디스크에 기록되는 동안 기다리지 않음을 지정합니다. NONBLOCKED 이벤트 모니터는 BLOCKED 이벤트 모니터 만큼 데이터베이스 작업

을 느리게 하지는 않습니다. 그러나, NONBLOCKED 이벤트 모니터는 활발히 활동 중인 시스템에서 데이터가 유실됩니다.

APPEND

이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 새로운 이벤트 데이터를 데이터 파일의 기존 스트림에 추가합니다. 이벤트 모니터가 다시 활성화될 경우, 꺼지지 않았던 것처럼 이벤트 파일에 다시 기록하기 시작합니다. APPEND는 기본 옵션입니다.

새로 작성된 이벤트 모니터가 아벤트 데이터를 기록할 디렉토리에 기존의 이벤트 데이터가 있을 경우, APPEND 옵션은 CREATE EVENT MONITOR 수행시 적용되지 않습니다.

REPLACE

이벤트 모니터가 작동될 때 이벤트 데이터 파일이 이미 존재할 경우, 그 이벤트 모니터가 모든 이벤트 파일을 지우고 00000000.evt 파일에 데이터를 기록하기 시작함을 지정합니다.

MANUALSTART

데이터베이스가 시작될 때 자동으로 이벤트 모니터가 시작되지 않음을 지정합니다. MANUALSTART 옵션을 사용하는 이벤트 모니터는 SET EVENT MONITOR STATE문을 사용하여 수동으로 활성화되어야 합니다. 기본 옵션입니다.

AUTOSTART

데이터베이스가 시작될 때 자동으로 이벤트 모니터가 시작됨을 지정합니다.

ON NODE

지정된 특정 파티션을 지시하는 키워드

노드 번호

이벤트 모니터가 수행되어 이벤트를 기록하는 파티션 번호를 지정합니다. GLOBAL로 정의된 모니터 영역으로, 모든 파티션은 지정된 파티

션 번호로 보고합니다. 입출력 구성요소는 지정된 파티션에서 실제로 수행되어 해당되는 레코드를 해당 파티션의 /tmp/dlocks 디렉토리에 기록합니다.

GLOBAL

이벤트 모니터는 모든 파티션으로부터 보고합니다. DB2 Universal Database 버전 7에서 파티션된 데이터베이스의 경우, 교착 상태 이벤트 모니터만을 GLOBAL로 정의할 수 있습니다. 글로벌 이벤트 모니터는 시스템의 모든 노드에 대해 교착 상태를 보고합니다.

LOCAL

이벤트 모니터는 수행중인 파티션에서만 보고합니다. 데이터베이스 활동에 대한 부분적 추적을 합니다. 이것은 기본값입니다.

규칙

- 이벤트 유형(DATABASE, TABLES, DEADLOCKs,...) 각각은 특정 이벤트 모니터 정의에 한번만 지정될 수 있습니다.

주

- 이벤트 모니터 정의는 SYSCAT.EVENTMONITORS 카탈로그 뷰에 기록됩니다. 이벤트 그 자체들은 SYSCAT.EVENTS 카탈로그 뷰에 기록됩니다.
- 데이터베이스 모니터의 사용과 파이프 및 파일의 데이터 해석에 관한 *시스템 모니터 안내 및 참조서*에서 자세한 정보를 참조하십시오.

예

예 1: 다음 예는 SMITHPAY라고 하는 이벤트 모니터를 작성합니다. 이 이벤트 모니터는 데이터베이스에 대해, 그리고 JSMITH 권한 부여 ID가 소유하는 PAYROLL 응용프로그램에서 수행되는 SQL문에 대해 이벤트 데이터를 수집합니다. 데이터는 절대 경로 /home/jsmith/event/smithpay/에 추가됩니다. 최대 25개의 파일이 작성됩니다. 각 파일은 최대 1 024 4K 페이지가 됩니다. 파일 I/O는 비블록화됩니다.

```
CREATE EVENT MONITOR SMITHPAY
   FOR DATABASE, STATEMENTS
   WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
```

```
    WRITE TO FILE '/home/jsmith/event/smithpay'
    MAXFILES 25
    MAXFILESIZE 1024
    NONBLOCKED
    APPEND
```

예 2: 다음 예는 DEADLOCKS_EVTS라고 하는 이벤트 모니터를 작성합니다. 이 이벤트 모니터는 교착 상태 이벤트를 수집하고 상대 경로 DLOCKS에 기록합니다. 파일 하나가 작성되고, 최대 파일 크기가 없습니다. 이벤트 모니터가 활성화될 때마다, 이벤트 데이터를 00000000.evt 파일(있는 경우)에 추가합니다. 이벤트 모니터는 데이터베이스가 시작될 때마다 시작됩니다. I/O는 기본적으로 블록화됩니다.

```
CREATE EVENT MONITOR DEADLOCK_EVTS
   FOR DEADLOCKS
   WRITE TO FILE 'DLOCKS'
   MAXFILES 1
   MAXFILESIZE NONE
   AUTOSTART
```

예 3: 다음 예는 DB_APPLS라고 하는 이벤트 모니터를 작성합니다. 이 이벤트 모니터는 연결 이벤트를 수집하고 데이터를 named pipe /home/jsmith/applpipe에 씁니다.

```
CREATE EVENT MONITOR DB_APPLS
   FOR CONNECTIONS
   WRITE TO PIPE '/home/jsmith/applpipe'
```

CREATE FUNCTION

이 명령문은 응용프로그램 서버(AS)에서 사용자 정의 함수나 함수 템플리트를 등록하거나 정의하는 데 사용됩니다.

이 명령문을 사용하여 작성될 수 있는 함수의 유형은 다섯 가지 입니다. 이들 각 함수가 별도로 설명됩니다.

- 외부 스칼라

  이 함수는 프로그래밍 언어로 작성되며 스칼라 값을 리턴합니다. 함수의 여러 가지 속성과 더불어 실행 가능한 외부 기능이 데이터베이스에 등록됩니다. 655 페이지의 『CREATE FUNCTION(외부 스칼라)』에서 참조하십시오.

- 외부 테이블

  이 함수는 프로그래밍 언어로 작성되며 완전한 테이블을 리턴합니다. 함수의 여러 가지 속성과 더불어 실행 가능한 외부 기능이 데이터베이스에 등록됩니다. 685 페이지의 『CREATE FUNCTION(외부 테이블)』에서 참조하십시오.

- OLE DB 외부 테이블

  사용자 정의 OLE DB 외부 함수는 OLE DB 공급자로부터 데이터를 액세스하기 위해 데이터베이스에 등록됩니다. 704 페이지의 『CREATE FUNCTION(OLE DB 외부 테이블)』에서 참조하십시오.

- 소스 또는 템플리트

  소스 함수는 이미 데이터베이스에 등록된 다른 함수(내장, 외부, SQL 또는 소스 함수 중 하나)를 호출하여 구현됩니다. 714 페이지의 『CREATE FUNCTION (소스 또는 템플리트)』에서 참조하십시오.

  어느 유형의 값이 리턴될지를 정의하나 실행 코드는 포함하지 않는 *함수 템플리트*라고 하는 부분적 함수를 작성할 수도 있습니다. 사용자는 이를 연합 시스템 내에 있는 데이터 소스 함수에 맵핑하여 연합 데이터베이스로부터 데이터 소스 함수가 호출될 수 없게 합니다. 함수 템플리트는 연합 서버라 지정되는 응용프로그램 서버(AS)에만 등록될 수 있습니다.

- SQL 스칼라, 테이블 또는 행

함수 내용은 SQL로 작성되고 데이터베이스에서 등록으로 함께 정의됩니다. 이 함수 내용은 스칼라 값, 테이블 또는 단일 행을 리턴합니다. 726 페이지의 『CREATE FUNCTION (SQL 스칼라, 테이블 또는 행)』에서 참조하십시오.

CREATE FUNCTION(외부 스칼라)

이 명령문은 응용프로그램 서버(AS)에 사용자 정의 외부 스칼라 함수를 등록하는 데 사용됩니다. *스칼라 함수*는 호출될 때마다 단일 값을 리턴하며, 일반적으로 SQL 표현식이 유효하면 이 함수도 유효합니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 함수의 스키마 이름이 기존의 스키마를 가리키지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 함수의 스키마 이름이 기존의 스키마를 가리키지 않을 경우, 데이터베이스에 대한 CREATEIN 특권

비분리 함수를 작성하려면, 명령문의 권한 부여 ID가 보유한 특권으로 다음 중 적어도 하나가 포함되어야 합니다.

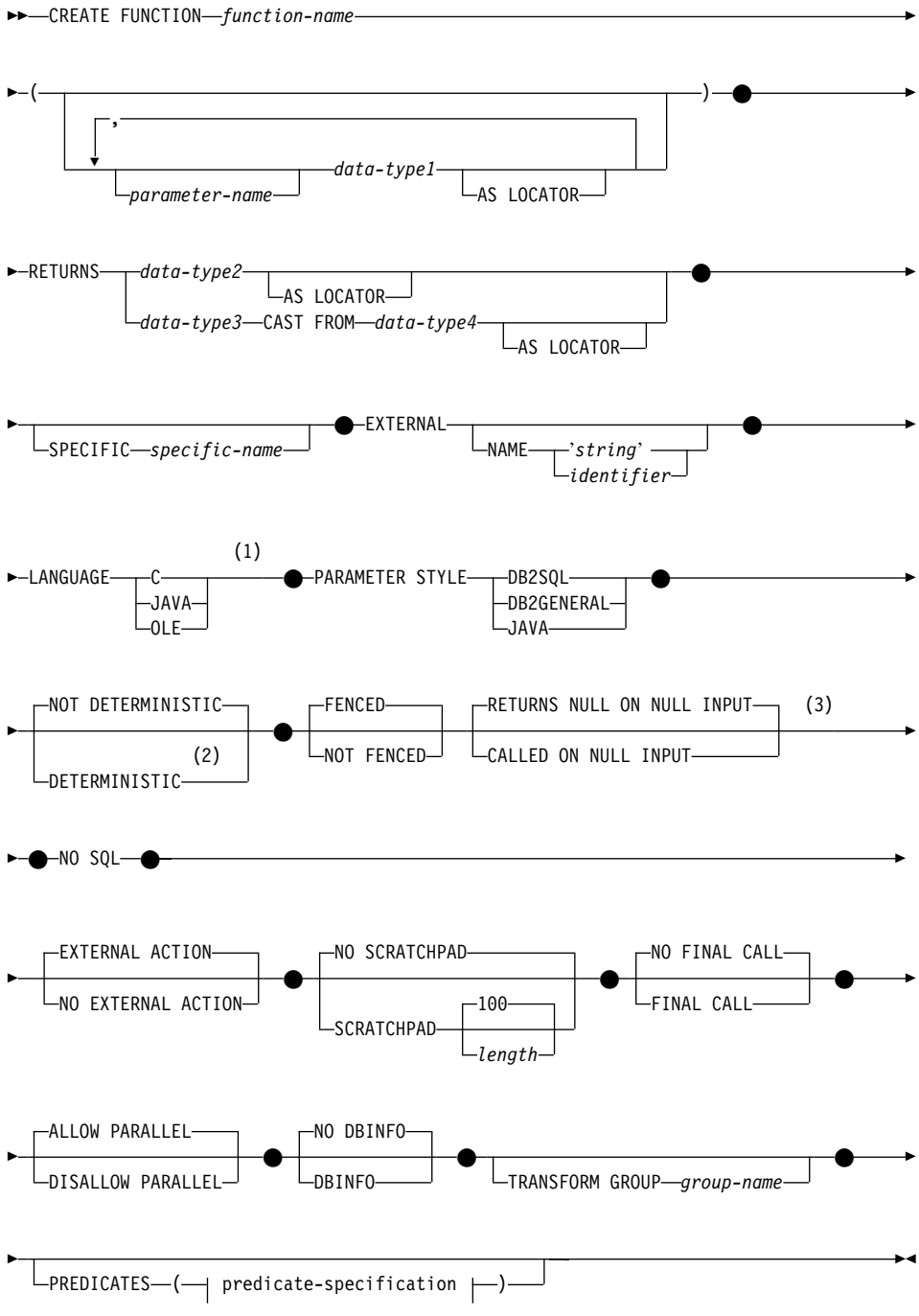
- 데이터베이스에서의 CREATE_NOT_FENCED 권한
- SYSADM 또는 DBADM 권한

범위가 정해진 함수를 작성하려면, 추가 권한이나 특권이 필요하지 않습니다.

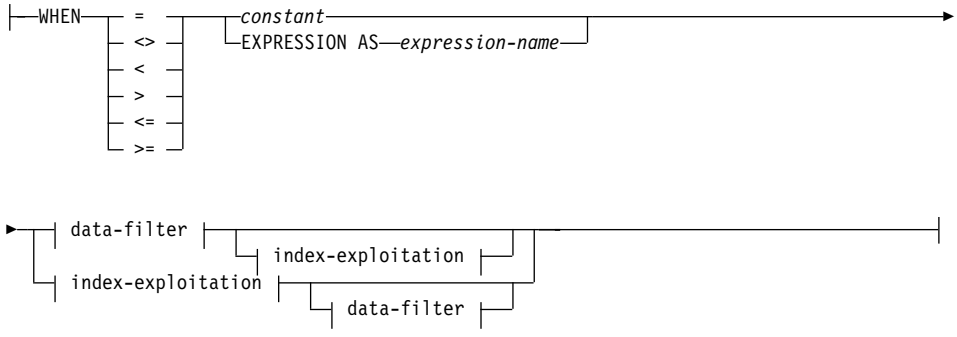
권한 부여 ID가 작업을 수행할 권한이 부족한 경우, 오류(SQLSTATE 42502)가 발생합니다.

구문

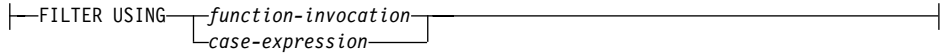
CREATE FUNCTION(외부 스칼라)



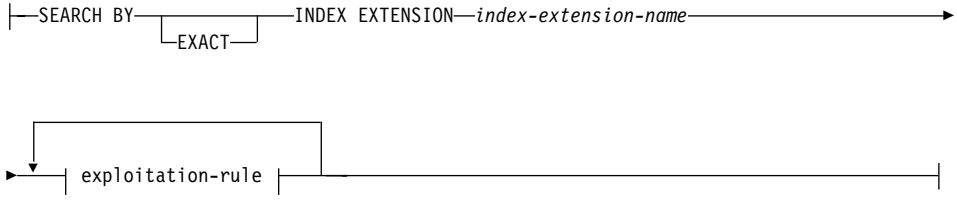
predicate-specification:



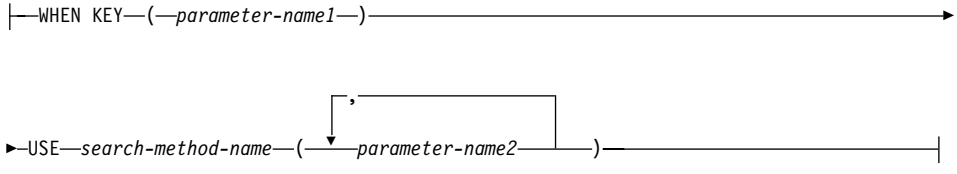
data-filter:



index-exploitation:



exploitation-rule:



주:

1 LANGUAGE SQL도 또한 지원됩니다. 726 페이지의 『CREATE FUNCTION (SQL 스칼라, 테이블 또는 행)』에서 참조하십시오.

2 DETERMINISTIC 대신 NOT VARIANT를 지정할 수 있으며, NOT DETERMINISTIC 대신 VARIANT를 지정할 수 있습니다.

3 CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있으며, RETURNS NULL ON NULL INPUT 대신 NOT NULL CALL을 지정할 수 있습니다.

설명

함수 이름

정의되는 함수를 명명합니다. 이 이름은 함수를 지시하는 규정화되거나 규정화되지 않는 이름입니다. *함수 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18인)입니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다. 규정된 이름은 첫번째 매개변수가 구조화 유형일 경우 그 첫번째 매개변수의 데이터 유형과 같으면 안됩니다.

매개변수 개수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고), 내재적 또는 명시적 규정자를 포함하여, 이름은 카탈로그에 설명된 함수나 메소드를 식별하지 않아야 합니다 (SQLSTATE 42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정화되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 된 이름이 지정되는 경우, *스키마 이름*은 "SYS"로 시작할 수 없습니다. 그렇지 않으면, 오류(SQLSTATE 42939)가 발생합니다.

술어에서 키워드로 사용되는 이름중 일부는 시스템에서 사용할 수 있도록 예약되어 있어서, *함수 이름*으로 사용할 수 없습니다. SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 등이 이러한 이름이며, 217 페이지의 『기본 술어』에 기술된 비교 연산자도 여기에 포함됩니다. 이 규칙을 지키지 못하면 오류가 발생됩니다(SQLSTATE 42939).

일반적으로, 함수 시그니처에 약간의 차이가 있으면, 둘 이상의 함수에 동일한 이름을 사용할 수 있습니다.

이것에 대한 제한이 없다고 하여도, 겹쳐쓸 의도가 없으면 외부 사용자 정의 테이블 함수는 내장 함수와 이름이 같으면 안됩니다. 동일한 인수와 내장 스

칼라 또는 컬럼 함수와 동일한 이름(예: LENGTH, VALUE, MAX)을 가지나 의미는 다른 함수를 제공하면, 동적 SQL문 또는 정적 SQL 적용업무를 다시 바인드할 때 오류가 발생할 수 있습니다. 즉, 적용업무가 실패하거나 더욱 나쁜 경우, 잘못된 결과를 제공하면서 성공적으로 수행된 것처럼 나타날 수도 있습니다.

매개변수 이름

후속 함수 정의에 사용될 수 있는 매개변수에 이름을 지정합니다. 매개변수 이름은 술어 스펙의 *색인 노출* 절에서 함수의 매개변수를 참조하기 위해 필요합니다.

(데이터 유형1,...)

함수의 입력 매개변수 개수를 식별하고, 각 매개변수의 데이터 유형을 지정합니다. 함수가 받기를 기대하는 각 매개변수에 대해 목록에서 한 항목만 지정할 수 있습니다. 매개변수 개수는 90개를 초과할 수 없습니다. 이 한계를 초과하면, 오류(SQLSTATE 54023)가 발생합니다.

매개변수가 없는 함수를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예:

```
CREATE FUNCTION WOOFER() ...
```

스키마 내에서 동일하게 명명된 두 함수는 모든 해당되는 매개변수에 대해 완전히 같은 유형을 가질 수 없습니다. 길이, 정밀도 및 스케일은 이 유형의 비교에서 고려되지 않습니다. 그러므로, CHAR(8) 및 CHAR(35)는 같은 유형으로 간주되고, DECIMAL(11,2) 및 DECIMAL (4,3)도 마찬가지입니다. DECIMAL 및 NUMERIC과 같이 이러한 목적을 위해 동일한 유형으로 처리되도록 하는 유형들이 더 있습니다. 시그니처가 중복되면 SQL 오류(SQLSTATE 42723)가 발생합니다.

예를 들어, 다음 명령문이 제공되면,

```
CREATE FUNCTION PART (INT, CHAR(15)) ...
CREATE FUNCTION PART (INTEGER, CHAR(40)) ...


CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

두 번째와 네번째 명령문이 중복되는 함수로 간주되므로 이들 명령문은 실패합니다.

데이터 유형1

매개변수의 데이터 유형을 지정합니다.

- CREATE TABLE문의 *데이터 유형1* 정의에 지정될 수 있고 함수를 작성하는 데 사용되는 언어에 해당 항목이 있는 SQL 데이터 유형 스펙과 약어를 지정할 수 있습니다. 사용자 정의 함수와 관련하여 SQL 데이터 유형과 호스트 언어 데이터 유형 사이의 맵핑에 관한 자세한 사항은 *응용프로그램 개발 안내서*의 해당 언어 부분을 참조하십시오.
- DECIMAL (및 NUMERIC)은 LANGUAGE C와 OLE에 대해 유효하지 않습니다(SQLSTATE 42815). DECIMAL의 대체 사용 방안에 대해서는 *응용프로그램 개발 안내서*에서 참조하십시오.
- REF(*유형 이름*)은 매개변수의 유형으로 지정할 수 있습니다. 그러나, 그러한 매개변수는 범위가 지정되지 않아야 합니다.
- 연관되는 변환 그룹에 적절한 변환 함수가 존재할 경우, 구조화 유형을 지정할 수도 있습니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 LOB 위치 지정자가 실제 값 대신 UDF에 전달됨을 나타냅니다. 이로써 UDF로 전달되는 바이트 수가 상당히 절약되며, 특히 값의 일부 몇 바이트만이 UDF와 관련있는 경우 성능에도 도움이 됩니다. UDF에서 LOB 위치 지정자를 사용하는 방법은 *응용프로그램 개발 안내서*에 설명되어 있습니다.

다음은 매개변수 정의에서 AS LOCATOR절을 사용하는 예입니다.

```
CREATE FUNCTION foo (CLOB(10M) AS LOCATOR, IMAGE AS LOCATOR)
                    ...
```

여기서는 IMAGE가 LOB 유형 중 하나에 기초한 구별 유형입니다.

인수 이송 목적의 경우, AS LOCATOR절이 아무 효과를 갖지 않습니다. 예에서, 유형은 각각 CLOB와 IMAGE로 간주되는 데, 이는 CHAR 또는 VARCHAR 인수를 첫번째 인수로 함수에 전달할 수 있

음을 의미합니다. 마찬가지로, AS LOCATOR는 함수 시그니처에 아무 영향을 미치지 않으며 이는 (a) "함수 차수"라는 프로세스에 의해 DML에서 참조될 경우와 (b) COMMENT ON이나 DROP과 같은 DDL에서 참조될 경우 함수를 서로 결합하는 데 사용됩니다. 사실상, 이 절은 COMMENT ON이나 DROP에서 다른 의미없이 사용되기도 하고 사용되지 않기도 합니다.

LOB 또는 LOB에 기초한 구별 유형 이외의 유형에 대해 AS LOCATOR가 지정되는 경우 오류(SQLSTATE 42601)가 발생합니다.

함수가 분리(fenced)된 경우, AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

RETURNS

이 필수 절은 함수의 출력을 식별합니다.

데이터 유형2

출력의 데이터 유형을 지정합니다.

이 경우, 함수 매개변수에 대해 *데이터 유형1*에 기술된 외부 함수의 매개변수에 대한 것과 동일한 고려사항이 적용됩니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 실제 값 대신 LOB 위치 지정자가 UDF로부터 전달됨을 나타냅니다.

데이터 유형3 **CAST FROM** *데이터 유형4*

출력의 데이터 유형을 지정합니다.

이 형태의 RETURNS절은 함수 코드에 의해 리턴된 데이터 유형에서 호출 명령문으로 서로 다른 데이터 유형을 리턴하는 데 사용됩니다. 예를 들어, 다음 명령문에서,

```
CREATE FUNCTION GET_HIRE_DATE(CHAR(6))
       RETURNS DATE CAST FROM CHAR(10)
                ...
```

함수 코드는 데이터베이스 관리 프로그램으로 CHAR(10)을 보낸 후, 이를 DATE로 변환하여 그 값을 호출 명령문에 전달합니다. *데이터 유형4*

는 *데이터 유형3* 매개변수로 변환될 수 있어야 합니다. 변환할 수 없는 경우, 오류(SQLSTATE 42880)가 발생됩니다(변환 가능에 대한 정의는 106 페이지의 『데이터 유형간의 변환』에서 참조하십시오.).

*데이터 유형3*의 길이, 정밀도 또는 스케일은 *데이터 유형4*에서 추론할 수 있으므로, 허용되더라도 *데이터 유형3*에 대한 매개변수 작성 유형에 대해 길이, 정밀도 또는 스케일을 지정하지 않아도 됩니다. 대신 빈 괄호가 사용됩니다(예: VARCHAR()가 사용될 수 있음). 매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용될 수 없습니다(SQLSTATE 42601).

구별 유형과 구조화 유형은 *데이터 유형4*에 지정되는 유형으로는 유효하지 않습니다(SQLSTATE 42815).

유형변환(cast) 조작도 변환 오류를 야기할 수 있는 런타임 점검사항입니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형인 *데이터 유형4* 스펙의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 실제 값 대신 LOB 위치 지정자가 UDF로부터 전달됨을 나타냅니다. UDF에서 LOB 위치 지정자를 사용하는 방법은 *응용프로그램 개발 안내서*에 설명되어 있습니다.

SPECIFIC *특정 이름*

정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. *특정 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18인)입니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다. 내재적 또는 명시적 규정자를 포함하여, 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스나 메소드 스펙을 식별하지 않아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*고유 이름*은 기존의 *함수 이름*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *함수 이름*에 대해 사용되었던 규정자가 사용됩니다. 규정자가 지정되면, *함수 이름*의 명시적 또는 암시적 규정자와 같아야 합니다. 그렇지 않으면 오류(SQLSTATE 42882)가 발생합니다.

*특정 이름*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

EXTERNAL

이 절은 CREATE FUNCTION문이 내부 프로그래밍 언어로 작성된 코드를 기초로 문서화된 연계 규칙과 인터페이스에 따라 새 함수를 등록하는 데 사용됩니다.

NAME절이 지정되지 않을 경우, "NAME *함수 이름*"이 사용됩니다.

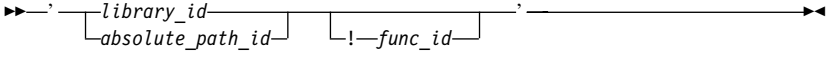
NAME *'문자열'*

이 절은 정의되는 함수를 구현하는 사용자 작성 코드의 이름을 식별합니다.

'문자열' 옵션은 최대 254 문자로 된 문자열 상수입니다. 문자열에 사용되는 형식은 지정되는 LANGUAGE에 따라 다릅니다.

- LANGUAGE C의 경우:

  지정된 *string*은 라이브러리 이름과 라이브러리 내의 함수입니다. 이것은 데이터베이스 관리자가 작성하여 사용자 정의 함수를 실행할 때 호출합니 다. CREATE FUNCTION문이 수행될 때 라이브러리(그리고 그 라이브러리 내의 함수)는 존재할 필요가 없습니다. 그러나 함수가 SQL 문에서 사용될 경우, 라이브러리와 그 라이브러리내의 함수가 존재해야 하고 데이터베이스 서버 머신에서 접근할 수 있어야 합니다. 그렇지 않으면 오류(SQLSTATE 42724)가 발생합니다.



  작은 따옴표 내에서 공백은 허용되지 않습니다.

  library_id

  함수를 포함하는 라이브러리 이름을 식별합니다. 데이터베이스 관리 프로그램은 .../sqllib/함수 디렉토리(UNIX용 시스템) 또는

...*instance_name*\함수 디렉토리(OS/2, 그리고 DB2INSTPROF 레지스트리 변수에 지정된 Windows 32 비트 운영 체제)에서 그 라이브러리를 찾습니다. 이곳은 데이터베이스 관리 프로그램을 수행하기 위해 사용되는 제어하는 sqllib 디렉토리를 데이터베이스 관리 프로그램이 찾는 곳입니다. 예를 들어, UNIX 기본 시스템의 제어 sqllib 디렉토리는 /u/$DB2INSTANCE/sqllib입니다.

'myfunc'가 UNIX 기반 시스템에서 *library_id*라면, 데이터베이스 관리 프로그램이 /u/production으로부터 수행되고 있는 경우 데이터베이스 관리 프로그램이 라이브러리 /u/production/sqllib/function/myfunc에서 함수를 찾게 할 것입니다.

OS/2 및 Windows 32 비트 운영 체제의 경우, 데이터베이스 관리 프로그램은 *library_id*가 함수 디렉토리에 위치되지 않은 경우에 LIBPATH 또는 PATH에서 찾습니다. OS/2에서, library_id는 8 문자를 초과해선 안됩니다.

absolute_path_id

함수를 포함하는 파일의 전체 경로 이름을 나타냅니다.

예를 들어, UNIX 기본 시스템에서 '/u/jchui/mylib/myfunc'는 데이터베이스 관리 프로그램이 /u/jchui/mylib 에서 myfunc 공유 라이브러리를 찾도록 합니다.

OS/2 및 Windows 32 비트 운영 체제에서, 'd:\mylib\myfunc'는 데이터베이스 관리 프로그램이 d:\mylib 디렉토리로부터 동적 링크 라이브러리인 myfunc.dll 파일을 로드하게 합니다. OS/2에서 이 스펙의 마지막 부분(즉, dll의 이름)은 8 문자를 초과해선 안 됩니다.

! func_id

호출될 함수의 입력점 이름을 식별합니다. !는 는 library id와 function id 사이의 분리 문자로 사용됩니다. *! func_id*를 생략하면, 데이터베이스 관리 프로그램은 라이브러리가 링크되었을 때 설정된 기본 진입점을 사용합니다.

예를 들면, UNIX 기본 시스템에서 'mymod!func8'은 데이터베이스 관리 프로그램이 라이브러리 $inst_home_dir/sqllib/function/mymod를 찾아 그 라이브러리 내의 진입점 func8을 사용하도록 지정합니다.

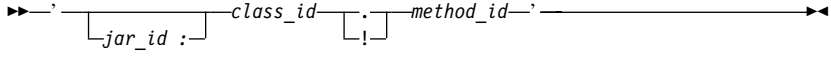
OS/2 및 Windows 32 비트 운영 체제에서, 'mymod!func8'은 데이터베이스 관리 프로그램이 mymod.dll 파일을 로드하고 동적 링크 라이브러리(DLL)에서 func8() 함수를 호출하게 합니다.

문자열이 적절히 형성되지 않으면, 오류(SQLSTATE 42878)가 발생합니다.

모든 외부 함수의 내용은 데이터베이스의 각 파티션에서 사용 가능한 디렉토리에 있어야 합니다.

- LANGUAGE JAVA의 경우:

  지정된 *문자열*에는 선택적 jar 파일 식별자, 클래스 식별자 및 메소드 식별자가 포함됩니다. 이는 작성되는 사용자 정의 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출하는 것입니다. CREATE FUNCTION문이 수행될 때 클래스 식별자와 방법 식별자는 존재하지 않아도 됩니다. *jar_id*를 지정할 경우, CREATE FUNCTION문이 실행될 때 존재해야 합니다. 그러나 함수가 SQL문에서 사용될 경우, 방법 식별자가 있어야 하며 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다. 그렇지 않으면, 오류(SQLSTATE 42724)가 발생합니다.



  작은 따옴표 내에서 공백은 허용되지 않습니다.

  jar_id

  데이터베이스에 설치될 때 jar 콜렉션에 부여된 jar 식별자를 식별합니다. 간단한 식별자 또는 스키마 규정화된 식별자일 수 있습니다. 'myJar' 및 'mySchema.myJar'가 그 예입니다.

  class_id

  Java 오브젝트의 클래스 식별자를 나타냅니다. 클래스가 패키지의 일부일 경우, 클래스 식별자 부분에 완전한 패키지 접두어, 예를 들

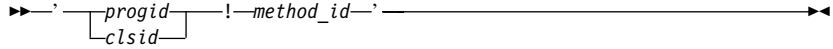
어, 'myPacks.UserFuncs'가 포함되어야 합니다. Java 가상 머신은 클래스를 디렉토리 '.../myPacks/UserFuncs/'에서 보게 됩니다. OS/2와 Windows 32 비트 운영 체제에서, Java 가상 머신은 디렉토리 '...\myPacks\UserFuncs\'에서 보게 됩니다.

method_id

호출할 Java 오브젝트의 방법 이름을 나타냅니다.

- LANGUAGE OLE의 경우:

  지정된 문자열은 OLE 프로그램가능 식별자(progid) 또는 클래스 식별자(clsid)와 방법 식별자로서, 작성중인 사용자 정의 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출한 것입니다. CREATE FUNCTION문이 수행될 때 프로그램가능 식별자, 클래스 식별자 및 방법 식별자는 존재하지 않아도 됩니다. 그러나 함수가 SQL문에서 사용될 경우, 방법 식별자가 있어야 하며 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다. 그렇지 않으면, 오류(SQLSTATE 42724)가 발생합니다.



  작은 따옴표 내에서 공백은 허용되지 않습니다.

  progid

  OLE 오브젝트의 프로그램가능 식별자를 나타냅니다.

  *progid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임 OLE API로 이송됩니다. 지정된 OLE 오브젝트는 작성 가능해야 하며 후기 바인딩(IDispatch형 바인딩이라고도 함)을 지원해야 합니다.

  clsid

  작성할 OLE 오브젝트의 클래스 식별자를 나타냅니다. OLE 오브젝트가 progid를 사용하여 등록되지 않은 경우 *progid*를 지정하기 위한 대안으로 사용할 수 있습니다. *clsid*은 다음과 같은 형식을 갖습니다.

  {nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}

여기서, 'n'은 영숫자 문자입니다. *clsid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임 OLE API로 이송됩니다.

method_id

호출할 OLE 오브젝트의 방법 이름을 나타냅니다.

NAME *식별자*

지정된 이 *식별자*는 SQL 식별자입니다. SQL 식별자는 문자열에서 *library id*로 사용됩니다. 분리 식별자가 아닐 경우, 식별자는 대문자로 겹쳐집니다. 식별자에 스키마 이름이 규정된 경우, 스키마 이름 부분은 무시됩니다. 이 형식의 이름은 LANGUAGE C에서만 사용할 수 있습니다.

LANGUAGE

이 필수 절은 사용자 정의 함수의 본체가 작성되는 언어 인터페이스 규약을 지정하는 데 사용됩니다.

C 즉, 데이터베이스 관리 프로그램은 C 함수처럼 사용자 정의 함수를 호출합니다. 사용자 정의 함수는 표준 ANSI C 프로토타입에 의해 정의된 대로 C 언어 호출 규칙과 연계 규칙에 따라야 합니다.

JAVA 이것은 데이터베이스 관리 프로그램이 Java 클래스의 한 방법으로 사용자 정의 함수를 호출하는 것을 의미합니다.

OLE 데이터베이스 관리 프로그램은 OLE 자동화 오브젝트에 의해 노출된 방법인 것처럼 사용자 정의 함수를 호출합니다. 사용자 정의 함수는 *OLE Automation Programmer's Reference*에 기술된 것처럼 OLE 자동화 데이터 유형 및 호출 메카니즘과 일치해야 합니다.

LANGUAGE OLE는 Windows 32 비트 운영 체제용 DB2에 저장된 사용자 정의 함수에 대해서만 지원됩니다

PARAMETER STYLE

이 절은 함수로부터 값을 리턴하고 매개변수를 전달하기 위한 규약을 지정하는 데 사용됩니다.

DB2SQL

이 절은 C 언어 호출 및 연결 규약에 부합하거나 OLE 자동화 오브젝트에 의해 노출된 방법인 외부 함수로 매개변수를 전달하고 외부 함

수의 값을 리턴하는 데 사용할 규약을 지정하는 데 사용됩니다. LANGUAGE C 또는 LANGUAGE OLE를 사용할 때 이를 지정해야 합니다.

DB2GENERAL

이 절은 Java 클래스에 방법으로 정의된 외부 함수로부터 값을 리턴하고 매개변수를 전달하는 데 사용할 규약을 지정하는 데 사용됩니다. 이것은 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

값 DB2GENRL는 DB2GENERAL의 동의어로 사용할 수 있습니다.

JAVA 이는 함수가 Java 언어 및 SQLJ 루틴 스펙을 준수하는 규약을 전달하는 매개변수를 사용할 것임을 의미합니다. 이것은 LANGUAGE JAVA가 사용되고 매개변수나 리턴 유형으로 구조화 유형이 없을 경우에만 지정할 수 있습니다(SQLSTATE 429B8). PARAMETER STYLE JAVA 함수는 FINAL CALL, SCRATCHPAD 또는 DBINFO 절을 지원하지 않습니다.

매개변수를 전달하는 방법에 대한 *응용프로그램 개발 안내서*에서 자세한 내용을 참조하십시오.

DETERMINISTIC 또는 **NOT DETERMINISTIC**

이 선택적 절은 함수가 항상 주어진 값에 대해 동일한 결과를 리턴하는지(DETERMINISTIC) 또는 함수가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 함수는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC를 지정하여 방지할 수 있습니다. NOT DETERMINISTIC 함수의 예는 난수(random-number) 생성 프로그램입니다. DETERMINISTIC 함수의 예는 입력의 제곱근을 판별하는 함수입니다.

FENCED 또는 **NOT FENCED**

이 절은 함수가 데이터베이스 관리 프로그램 운영 환경의 프로세스나 주소 공간에서 수행하기에 "안전한"지(NOT FENCED) 않은지(FENCED)를 지정합니다.

함수가 FENCED로 등록되면, 데이터베이스 관리 프로그램은 함수에서 액세스를 못하도록 내부 자원(예를 들어, 데이터 버퍼)을 분리시킵니다. 대부분의 함수는 FENCED나 NOT FENCED로 수행되는 옵션을 갖고 있습니다. 일반적으로, FENCED로 수행되는 함수는 NOT FENCED로 수행되는 것과 유사하게 수행되지 않습니다.

경고: 적절히 코딩, 검토 및 테스트되지 않은 함수에 대해 NOT FENCED를 사용하면 DB2의 무결성을 보완할 수 있습니다. DB2는 발생할 수도 있는 많은 공통 유형의 실수에 대한 예방책을 취하나, NOT FENCED 사용자 정의 함수가 사용될 때 완전한 무결성을 보증할 수 없습니다.

FENCED를 사용하면 NOT FENCED를 사용할 때보다 데이터베이스 무결성에 대해 더 나은 보호 수준을 제공하지만, 적절하게 코딩, 검토 및 테스트되지 않은 FENCED UDF는 DB2 장애를 야기할 수도 있습니다.

대부분의 사용자 정의 함수는 FENCED나 NOT FENCED로 수행될 수 있어야 합니다. LANGUAGE OLE에서는 함수에 대해 FENCED만을 지정할 수 있습니다(SQLSTATE 42613).

함수가 분리(fenced)된 경우, AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

FENCED에서 NOT FENCED로 변경하려면, 함수는 다시 등록되어야 합니다(먼저 그 함수를 삭제한 후 다시 작성). 사용자 정의 함수를 NOT FENCED로 등록하려면 SYSADM 권한, DBADM 권한 또는 특수 권한(CREATE_NOT_FENCED)이 필요합니다.

RETURNS NULL ON NULL INPUT 또는 CALLED ON NULL INPUT

이 선택적 절은 널(NULL) 값인 인수가 있을 경우 외부 함수로의 호출을 방지하기 위해 사용됩니다. 사용자 정의 함수(UDF)가 아무 매개변수도 가지지 않는 것으로 정의되면, 물론 이러한 널(NULL) 인수 상태가 발생할 수 없으며 이 스펙의 코딩 방법이 문제가 되지 않습니다.

RETURNS NULL ON NULL INPUT이 지정되고 실행 시 함수 인수 중 어느 하나가 널(NULL)일 경우, 사용자 정의 함수는 호출되지 않으며 결과는 널(NULL) 값이 됩니다.

CALLED ON NULL INPUT이 지정될 경우, 어느 인수가 널(NULL)인지에 관계 없이 사용자 정의 함수가 호출됩니다. 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 그러나, 널(NULL) 값 인수 값에 대한 테스트 책임은 UDF에 있습니다.

NULL CALL이 역방향 및 계열 호환성을 위해 CALLED ON NULL INPUT에 대한 동의어로서 사용될 수도 있습니다. 마찬가지로, NOT NULL CALL이 RETURNS NULL ON NULL INPUT에 대한 동의어로서 사용될 수도 있습니다.

NO SQL

이 필수 절은 함수가 SQL문을 실행할 수 없음을 나타냅니다. 이를 수행하면 런타임 오류(SQLSTATE 38502)가 발생합니다.

NO EXTERNAL ACTION 또는 **EXTERNAL ACTION**

이 선택적 절은 데이터베이스 관리 프로그램에 의해 관리되지 않는 오브젝트의 상태를 변경하는 일부 조치를 함수에서 사용하고 있는 지를 지정합니다. 외부 영향이 없는 함수를 가정하는 최적화는 EXTERNAL ACTION을 지정하여 금지됩니다. 예를 들어, 메시지 전송, 벨 울림 또는 파일에 레코드 기록.

NO SCRATCHPAD 또는 **SCRATCHPAD** *길이*

이 선택적 절은 외부 함수에 대해 스크래치패드가 제공되는지의 여부를 지정할 때 사용할 수 있습니다(사용자 정의 함수는 재입력이 가능하도록 적극 권장되므로, 스크래치패드는 함수가 한 호출에서 다음 호출로의 "상태를 저장" 하기 위한 수단을 제공합니다.).

SCRATCHPAD가 지정되면, 사용자 정의 함수를 처음 호출할 때 외부 함수에서 사용할 스크래치패드에 메모리가 할당됩니다. 이 스크래치패드는 다음 특성을 갖습니다.

- *길이*를 지정할 경우, 이것은 스크래치 패드의 *크기*를 바이트 단위로 설정합니다. 이 값은 1 - 32 767 사이여야 합니다(SQLSTATE 42820). 기본값 *크기*는 100바이트입니다.
- 모든 X'00'에 대해 초기화됩니다.
- 범위는 SQL문입니다. SQL문에서 외부 함수에 대해 참조당 하나의 스크래치패드가 있습니다. 그래서, 다음 명령문의 UDFX 함수가 SCRATCHPAD 키워드를 사용하여 정의한 경우, 세 개의 스크래치패드가 지정됩니다.

```
SELECT A, UDFX(A) FROM TABLEB
  WHERE UDFX(A) > 103 OR UDFX(A) < 19
```

ALLOW PARALLEL이 지정되거나 기본 설정되면, 범위가 위와 달라집니다. 함수가 복수 파티션에서 실행될 경우, SQL문에서의 각 함수 참조에 대해 스크래치패드는 함수가 처리되는 각 파티션에 지정됩니다. 마찬가지로, 파티션 내 병렬 처리를 사용하여 조회를 실행할 경우, 네 개 이상의 스크래치패드를 지정할 수도 있습니다.

- 지속적입니다. 그 내용이 하나의 외부 함수 호출에서 다음 호출로 보존됩니다. 한 호출상의 외부 함수에서 변경한 스크래치패드는 다음 호출에도 있게 됩니다. 데이터베이스 관리 프로그램이 각 SQL문 실행의 시작시에 스크래치패드를 초기 설정합니다. 데이터베이스 관리 프로그램이 각 부속 조회 실행의 시작시에 스크래치패드를 재설정합니다. FINAL CALL 옵션이 지정되면, 시스템에서 스크래치패드를 재설정하기 전에 최종 호출을 발행합니다.
- 외부 함수가 획득할 수 있는 시스템 자원(예: 메모리)에 대해 중심점으로 사용됩니다. 함수는 첫번째 호출에서 메모리를 획득하고, 그 주소를 스크래치패드에 보관한 후 다음 호출에서 이를 참조할 수 있습니다.

  (시스템 자원을 확보한 경우에는 FINAL CALL 키워드를 지정해야 합니다. 그러면, 명령문 끝에서 특수 호출이 이루어져 외부 함수가 확보한 시스템 자원이 해제됩니다.)

SCRATCHPAD가 지정되면, 사용자 정의 함수를 호출할 때마다 추가 인수가 스크래치패드가 주소지정된 외부 함수로 전달됩니다.

NO SCRATCHPAD가 지정되면, 어떤 스크래치패드도 외부 함수에 할당되거나 전달되지 않습니다.

SCRATCHPAD는 PARAMETER STYLE JAVA 함수에 지원되지 않습니다.

NO FINAL CALL 또는 **FINAL CALL**

이 선택적 절은 외부 함수에 대해 마지막 호출이 수행되는 지의 여부를 지정할 때 사용할 수 있습니다. 마지막 호출의 목적은 외부 함수가 획득한 시스템 자원을 해제할 수 있도록 하는 것입니다. 이것은 외부 함수가 메모리와 같은

시스템 자원을 확보하여 이들을 스크래치패드에 고정시키는 상황에서 SCRATCHPAD 키워드와 함께 사용하면 유용할 수 있습니다. FINAL CALL이 지정되면, 실행시

- 호출의 유형을 지정하는 추가 인수가 외부 함수에 전달됩니다. 호출 유형은 다음과 같습니다.
  - 정상 호출: SQL 인수가 전달되고 결과 리턴이 예상됩니다.
  - 첫번째 호출: 이 SQL문에서 사용자 정의 함수(UDF)에 대한 이 참조의 외부 함수에 대한 첫번째 호출. 첫번째 호출은 정상적인 호출입니다.
  - 마지막 호출: 함수가 자원을 해제할 수 있도록 하는 외부 함수에 대한 마지막 호출. 마지막 호출은 정상적인 호출이 아닙니다. 이 마지막 호출은 다음 상황에서 발생합니다.
    - 명령문의 끝: 이 경우는 커서가 커서 지향 명령문에 대해 닫혀 있거나, 명령문이 다른 실행을 통할 때 발생합니다.
    - 트랜잭션의 끝: 이 경우는 정상적인 명령문 끝이 발생하지 않을 때 발생됩니다. 예를 들어, 응용프로그램의 논리는 어떤 이유로 커서 닫기를 생략할 수 있습니다.

    WITH HOLD로 정의된 커서가 열려 있는 동안 확약 조작이 발생하면, 그 다음의 커서를 닫을 때 또는 응용프로그램 종료시 최종 호출이 수행됩니다.

  NO FINAL CALL이 지정되면 "호출 유형" 인수가 외부 함수로 전달되지 않으며 최종 호출도 이루어지지 않습니다.

  오류 발생시 이들 호출의 스칼라 UDF 처리 설명은 *응용프로그램 개발 안내서*에 포함됩니다.

FINAL CALL은 PARAMETER STYLE JAVA 함수에 지원되지 않습니다.

ALLOW PARALLEL 또는 **DISALLOW PARALLEL**

이 선택적 절은 함수에 대한 단일 참조에 대해 함수의 호출이 병렬화될 수 있는지의 여부를 지정합니다. 일반적으로, 대부분의 스칼라 함수 호출은 병렬화가 가능해야 하지만, 병렬화할 수 없는 함수(스크래치패드의 단일본에 근거한 함수)도 있을 수 있습니다. 스칼라 함수에 대해 ALLOW PARALLEL 또는

DISALLOW PARALLEL가 지정되는 경우, DB2는 이 스펙을 승인합니다. 함수에 적합한 키워드를 결정할 때 다음 사항을 고려해야 합니다.

- 모든 UDF 호출이 서로 완전히 독립되어 있는가? 그럴 경우, ALLOW PARALLEL을 지정하십시오.
- 각 UDF 호출이 스크래치패드를 갱신하여 다음 호출과 관련한 값을 제공하는가? (예를 들어, 계산기의 증분) 그럴 경우, DISALLOW PARALLEL을 지정하거나 기본값을 승인하십시오.
- 하나의 파티션에서만 발생해야 하는, UDF에 의해 수행되는 일부 외부 조치가 있습니까? 그럴 경우, DISALLOW PARALLEL을 지정하거나 기본값을 승인하십시오.
- 비용이 많이 드는 초기설정 처리를 최소한의 횟수로 수행할 수 있도록 스크래치패드가 사용되었는가? 그럴 경우, ALLOW PARALLEL을 지정하십시오.

어떤 경우, 모든 외부 함수의 내용이 데이터베이스의 모든 파티션에서 사용 가능한 디렉토리에 있어야 합니다.

구문 도표는 기본값이 ALLOW PARALLEL임을 나타냅니다. 단, 다음 옵션 중 하나 이상이 명령문에 지정되어 있을 경우 기본값은 DISALLOW PARALLEL입니다.

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

NO DBINFO 또는 **DBINFO**

이 선택적 절은 DB2가 확인한 특정 정보가 추가 호출 인수로 UDF에 전달되는지(DBINFO) 않은지(NO DBINFO)의 여부를 지정합니다. NO DBINFO가 기본값입니다. DBINFO는 LANGUAGE OLE(SQLSTATE 42613) 또는 PARAMETER STYLE JAVA에서는 지원되지 않습니다.

DBINFO가 지정되는 경우, 다음 정보가 들어 있는 UDF로 구조가 전달됩니다.

- 데이터베이스 이름 - 현재 연결된 데이터베이스의 이름.

- 응용프로그램 ID - 데이터베이스로의 각 연결에 형성되는 고유한 응용프로그램 ID.
- 응용프로그램 권한 부여 ID - 이 UDF와 응용프로그램 사이의 중첩 UDF에 관계없이 응용프로그램 런타임 권한 부여 ID.
- 코드 페이지 - 데이터베이스 코드 페이지를 식별합니다.
- 스키마 이름 - 테이블 이름과 완전히 같은 조건하에서 스키마의 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 테이블 이름 - UDF 참조가 UPDATE문의 SET절 오른쪽에 있거나 INSERT문의 VALUES 목록에 있는 항목인 경우에만 갱신 또는 삽입중인 테이블의 규정화되지 않은 이름이 들어갑니다. 그렇지 않으면 공백입니다.
- 컬럼 이름 - 테이블 이름과 완전히 같은 조건하에서, 갱신되거나 삽입중인 컬럼의 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 데이터베이스 버전/릴리스 - UDF를 호출한 데이터베이스 서버의 버전, 릴리스 및 개정 레벨을 나타냅니다.
- 플랫폼 - 서버의 플랫폼 유형을 포함합니다.
- 테이블 함수 결과 컬럼 수 - 외부 스칼라 함수에는 적용되지 않음.

구조와 이 구조가 사용자 정의 함수에 전달되는 방식에 대한 *응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.

TRANSFORM GROUP *그룹 이름*

함수를 호출할 때 사용자가 정의하는 구조화 유형변환에 대해 사용되는 변환 그룹을 나타냅니다. 변환은 함수 정의가 매개변수로 사용자 정의 구조화 유형을 포함하거나 데이터 유형을 리턴할 경우에 필요합니다. 이 절을 지정하지 않으면, 기본 그룹 이름인 DB2_FUNCTION이 사용됩니다. 지정된 (또는 기본값인) *그룹 이름*이 참조된 구조화 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42741). 필수 FROM SQL 또는 TO SQL 변환 함수가 제공된 그룹 이름과 구조화 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42744).

지시되었는지, 또는 내재되었는지에 따라 변환 함수 FROM SQL 및 TO SQL은 구조화 유형 및 해당되는 내장 유형 속성 사이에 적절하게 변환하는 SQL 함수여야 합니다.

PREDICATES

이 함수가 술어에서 사용될 경우 수행되는 필터링 또는 색인 확장 노출을 정의합니다. 술어 스펙은 검색 조건의 선택적 SELECTIVITY 절을 지정할 수 있도록 허용합니다. PREDICATES 절을 지정할 경우, 함수는 NO EXTERNAL ACTION의 DETERMINISTIC으로 정의되어야 합니다 (SQLSTATE 42613).

WHEN *비교 연산자*

비교 연산자("=", "<", ">", ">=", "<=", "<>")가 있는 술어에서 함수의 특정 사용을 소개합니다.

상수

데이터 유형이 함수의 RETURNS 유형에 호환 가능한 상수 값을 지정합니다(SQLSTATE 42818). 술어가 같은 비교 연산자와 이 상수를 사용하여 이 함수를 사용할 때, 지정된 필터링과 색인 노출이 최적화 알고리즘으로 간주됩니다.

EXPRESSION AS *표현식 이름*

표현식에 대한 이름을 제공합니다. 술어가 같은 비교 연산자와 표현식을 사용하여 이 함수를 사용할 때, 필터링과 색인 노출을 사용할 수도 있습니다. 표현식에는 검색 함수 인수로 사용될 수 있도록 표현식 이름이 지정됩니다. *표현식 이름*은 작성되는 함수의 *매개변수 이름*과 같을 수 없습니다(SQLSTATE 42711). 표현식을 지정할 경우, 그 표현식의 유형이 식별됩니다.

FILTER USING

결과 테이블의 추가 필터링에 외부 함수나 CASE 표현식을 지정할 수 있게 합니다.

함수 호출

결과 테이블의 추가 필터링을 수행하기 위해 사용할 수 있는 필터 함수를 지정합니다. 이것은 행 규정 여부를 판별하기 위해 사용자 정의 술어가 실행되어야 하는 행 수를 감소하는 정의된 함수(술어에서 사용되는)의 버전입니다. 색인에 의해 생성되는 결과가 사용자 정의 술

어에 대해 예상된 결과에 근접할 경우, 필터링 함수를 적용하는 것은 불필요할 수도 있습니다. 지정하지 않으면, 데이터 필터링은 수행되지 않습니다.

이 함수는 *매개변수 이름*, *표현식 이름* 또는 상수를 인수로 사용하고 (SQLSTATE 42703), 정수를 리턴합니다(SQLSTATE 428E4). 리턴 값 1은 행이 보존됨을 의미하고, 그렇지 않으면 행은 버려집니다.

이 함수는 또한 다음을 만족해야 합니다.

- LANGUAGE SQL로 정의하지 않아야 합니다(SQLSTATE 429B4).
- NOT DETERMINISTIC 또는 EXTERNAL ACTION으로 정의하지 않아야 합니다(SQLSTATE 42845).
- 매개변수 중 어느 하나의 데이터 유형으로 구조화 데이터 유형을 가지고 있지 않아야 합니다(SQLSTATE 428E3)
- 부속 조회를 포함하고 있지 않아야 합니다(SQLSTATE 428E4).

인수가 다른 함수나 메소드를 호출할 경우 중첩된 함수나 메소드에 대해 위의 네 가지 규칙도 시행됩니다. 그러나, 시스템에서 생성되는 observer 메소드는 인수가 내장 데이터 유형을 평가하면 필터 함수(또는 인수로 사용되는 함수나 메소드)에 대한 인수로 허용됩니다.

CASE-표현식

결과 테이블의 추가 필터링에 대한 CASE 표현식을 지정합니다. *searched-when-clause* 및 *simple-when-clause*는 *매개변수 이름*, *표현식 이름* 또는 상수를 사용할 수 있습니다(SQLSTATE 42703). FILTER USING *함수* 호출에 규칙이 지정된 외부 함수를 결과 표현식으로 사용할 수 있습니다. CASE 표현식에서 참조되는 함수나 메소드는 *함수* 호출 아래에 나열된 네 가지 규칙도 따라야 합니다.

부속 조회는 *CASE 표현식* 어디에서나 사용할 수 있습니다 (SQLSTATE 428E4).

CASE 표현식은 정수를 리턴해야 합니다(SQLSTATE 428E4). 결과 표현식에서 리턴 값 1은 행이 보존됨을 의미하고, 그렇지 않으면 행은 버려집니다.

색인 노출

색인을 노출하기 위해 사용할 수 있는 색인 확장의 검색 메소드 측면에서 규칙 세트를 정의합니다.

SEARCH BY INDEX EXTENSION *색인 표현식 이름*

색인 확장을 식별합니다. *색인 표현식 이름*은 기존의 색인 확장을 식별해야 합니다.

EXACT

색인 찾아보기는 술어 평가에 관하여 정확함을 나타냅니다. 원래 사용자 정의 술어 함수나 필터가 색인 찾아보기 후에 적용되지 않도록 DB2에 지시하려면 EXACT를 사용하십시오. EXACT 술어는 색인 찾아보기가 술어와 같은 결과를 리턴할 때 유용합니다.

EXACT를 지정하지 않을 경우, 원래의 사용자 정의 술어는 색인 찾아보기 후에 적용됩니다. 색인이 술어의 예측만을 제공할 것으로 예상되면, EXACT 옵션을 지정하지 마십시오.

색인 찾아보기가 사용되지 않으면, 필터 함수와 원래의 술어가 적용되어야 합니다.

노출 규칙

검색 목표 및 검색 인수, 그리고 이들을 사용하여 색인 확장에서 정의된 검색 메소드를 통해 색인 검색을 수행하는 방법에 대해 설명합니다.

WHEN KEY(*매개변수 이름1*)

이것은 탐색 목표를 정의합니다. 하나의 키에 대해 하나의 검색 목표만 지정할 수 있습니다. *매개변수 이름1* 값은 정의된 함수의 매개변수 이름을 식별합니다(SQLSTATE 42703 또는 428E8).

*매개변수 이름1*의 데이터 유형은 색인 확장에 지정된 소스 키의 데이터 유형과 일치해야 합니다(SQLSTATE 428EY). 내장 및 구별 데이터 유형에 대해, 그리고 구조화 유형에 대한 동일한 구조화 유형 계층 내에서 정확하게 일치해야 합니다.

이 절은 명명된 매개변수의 값이 지정된 색인 확장을 기초로 색인에서 다뤄지는 컬럼들일 경우에 올바르게 됩니다.

USE *검색 메소드 이름*(*매개변수 이름2,...*)

이것은 탐색 인수를 정의합니다. 이것은 색인 확장에 정의된 검색 메소드에서 사용할 검색 메소드를 식별합니다. *검색 메소드 이름*은 색인 확장에 정의된 검색 메소드와 일치해야 합니다(SQLSTATE 42743). *매개변수 이름2* 값은 EXPRESSION AS 절에서 정의된 함수의 매개변수 이름이나 *표현식 이름*을 식별합니다. 이것은 검색 목표에 지정된 매개변수 이름과 달라야 합니다(SQLSTATE 428E9). 매개변수 개수와 각 *매개변수 이름2*의 데이터 유형은 색인 확장에서 검색 메소드에 대해 정의된 매개변수와 일치해야 합니다. 내장 및 구별 데이터 유형에 대해, 그리고 구조화 유형에 대한 동일한 구조화 유형 계층 내에서 정확하게 일치해야 합니다.

주

- 하나의 데이터 유형이 다른 데이터 유형으로 변환 가능한 지를 판별할 때 CHAR 및 DECIMAL과 같은 매개변수화된 데이터 유형에 대해 길이나 정밀도, 그리고 스케일은 고려되지 않습니다. 그러므로, 함수를 사용할 때 원시 데이터 유형의 값을 목표 데이터 유형의 값으로 변환하려고 하면 오류가 발생할 수 있습니다. 예를 들어, VARCHAR은 DATE로 변환 가능하나 원시 유형이 실제로 VARCHAR(5)로 정의되면, 함수를 사용할 때 오류가 발생합니다.
- 사용자 정의 함수의 매개변수에 대한 데이터 유형을 선택할 때, 그 입력 값에 영향을 주게 될 승격 규칙을 고려하십시오(104 페이지의 『데이터 유형의 승격』 참조). 예를 들어, 입력 값으로 사용된 상수가 예상한 것과 다른 내장 데이터 유형을 가질 수도 있고, 좀더 심각하게는 예상한 데이터 유형으로 승격되지 않을 수 있습니다. 승격 규칙에 따라, 매개변수에 다음의 데이터 유형을 사용하는 것이 좋습니다.
  - SMALLINT 대신 INTEGER
  - REAL 대신 DOUBLE
  - CHAR 대신 VARCHAR
  - GRAPHIC 대신 VARGRAPHIC
- 여러 플랫폼에서의 UDF 이식성을 위해 다음과 같은 데이터 유형을 사용해서는 안됩니다.
  - FLOAT- DOUBLE 또는 REAL을 대신 사용

  - NUMERIC- DECIMAL을 대신 사용
  - LONG VARCHAR- CLOB(또는 BLOB)를 대신 사용
- 함수와 메소드는 겹치는 관계에 있지 않을 수도 있습니다(SQLSTATE 42745). 겹쳐쓰기에 대해서는 893 페이지의 『CREATE TYPE(구조화)』에서 자세한 내용을 참조하십시오.
- 함수는 메소드와 같은 시그니처를 가질 수 없습니다(함수의 첫번째 *매개변수 유형*을 메소드의 *주제 유형*과 비교하여)(SQLSTATE 42723).
- 외부 사용자 정의 함수의 작성, 컴파일 및 링크에 대한 *응용프로그램 개발 안내서*에서 정보를 참조하십시오.
- 아직 존재하지 않는 스키마 이름을 사용하여 함수를 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.

예

예 1: Pellow가 자신의 PELLOW 스키마에 CENTRE 함수를 등록하고 있습니다. 해당 키워드들을 기본값으로 하고 시스템이 함수 특정 이름을 제공한다고 가정합니다.

```
  CREATE FUNCTION CENTRE (INT,FLOAT)
      RETURNS FLOAT
      EXTERNAL NAME 'mod!middle'
      LANGUAGE C
      PARAMETER STYLE DB2SQL
DETERMINISTIC
NO SQL
      NO EXTERNAL ACTION
```

예 2 : 이제, McBride(DBADM 권한 보유)가 다른 CENTRE 함수를 PELLOW 스키마에 등록하여, 후속 DDL용으로 명시적인 특정 이름을 제공하고 명시적으로 모든 키워드 값을 제공합니다. 이 함수는 스크래치패드를 사용하며 후속되는 결과에 영향을 주는 데이터를 누산합니다. DISALLOW PARALLEL가 지정되었으므로 함수에 대한 모든 참조는 병렬화되지 않습니다. 따라서 한번만 초기화를 수행하고 그 결과를 저장하는 데 단일 스크래치패드가 사용됩니다.

```
CREATE FUNCTION PELLOW.CENTRE (FLOAT, FLOAT, FLOAT)
    RETURNS DECIMAL(8,4) CAST FROM FLOAT
    SPECIFIC FOCUS92
    EXTERNAL NAME 'effects!focalpt'
    LANGUAGE C   PARAMETER STYLE DB2SQL
    DETERMINISTIC   FENCED   NOT NULL CALL   NO SQL   NO EXTERNAL ACTION
    SCRATCHPAD   NO FINAL CALL
    DISALLOW PARALLEL
```

예 3: 다음은 규칙을 수행하기 위해 작성된 C 언어의 사용자 정의 함수 프로그램입니다.

```
output = 2 * input - 4
```

입력이 널(NULL)인 경우 널(NULL)을 리턴합니다. 이것은 CREATE FUNCTION 문이 NOT NULL CALL을 사용한 경우 더 간단하게 작성될 수 있습니다(즉, 널(NULL) 점검 없이). 사용자 정의 함수(UDF) 프로그램의 추가 예는 *응용프로그램 개발 안내서*에 나와 있습니다. CREATE FUNCTION문:

```
CREATE FUNCTION ntest1 (SMALLINT)
   RETURNS SMALLINT
   EXTERNAL NAME 'ntest1!nudft1'
   LANGUAGE C    PARAMETER STYLE DB2SQL
   DETERMINISTIC   NOT FENCED   NULL CALL
   NO SQL   NO EXTERNAL ACTION
```

프로그램 코드:

```
#include "sqlsystm.h"
/* NUDFT1 IS A USER_DEFINED SCALAR FUNCTION */
/* udft1 accepts smallint input
   and produces smallint output
   implementing the rule:
     if (input is null)
        set output = null;
   else
        set output = 2 * input - 4;
*/
void SQL_API_FN nudft1
     (short *input,      /* ptr to input arg */
      short *output,     /* ptr to where result goes */
      short *input_ind,  /* ptr to input indicator var */
      short *output_ind, /* ptr to output indicator var */
      char sqlstate[6],  /* sqlstate, allows for null-term */
      char fname[28],    /* fully qual func name, nul-term */
      char finst[19],    /* func specific name,  null-term */
```

```
      char msgtext[71])  /* msg text buffer,     null-term */
{
   /* first test for null input */
   if (*input_ind == -1)
   {
      /* input is null, likewise output */
      *output_ind = -1;
   }
   else
   {
      /* input is not null.  set output to 2*input-4 */
      *output = 2 * (*input) - 4;
      /* and set out null indicator to zero */
      *output_ind = 0;
   }
   /* signal successful completion by leaving sqlstate as is */
   /* and exit */
   return;
}
/* end of UDF: NUDFT1 */
```

예 4: 다음은 문자열에서 첫번째 모음의 위치를 리턴하는 Java UDF를 등록합니다. Java에 기록된 UDF는 분리된 채로 수행되며 클래스 javaUDF의 findvwl 방법입니다.

```
CREATE FUNCTION findv ( CLOB(100K))
     RETURNS INTEGER
     FENCED
     LANGUAGE JAVA
     PARAMETER STYLE JAVA
     EXTERNAL NAME 'javaUDFs.findvwl'
     NO EXTERNAL ACTION
     CALLED ON NULL INPUT
     DETERMINISTIC
     NO SQL
```

예 5: 이 예는 유형 SHAPE의 두 매개변수인 g1 및 g2를 입력으로 사용하는 사용자 정의 술어 WITHIN에 대한 윤곽을 보여줍니다.

```
CREATE FUNCTION within (g1 SHAPE, g2 SHAPE)
     RETURNS INTEGER
     LANGUAGE C
     PARAMETER STYLE DB2SQL
     NOT VARIANT
     NOT FENCED
     NO SQL
     NO EXTERNAL ACTION
     EXTERNAL NAME 'db2sefn!SDESpatilRelations'
```

```
PREDICATES
WHEN = 1
FILTER USING mbrOverlap(g1..xmin, g1..ymin, g1..xmax, g1..max,
                        g2..xmin, g2..ymin, g2..xmax, g2..ymax)
SEARCH BY INDEX EXTENSION gridIndex
WHEN KEY(g1) USE withinExplRule(g2)
WHEN KEY(g2) USE withinExplRule(g1)
```

WITHIN 함수의 설명은 사용자 정의 함수의 설명과 유사하지만, 다음의 추가사항은 이 함수가 사용자 정의 술어에서 사용될 수 있음을 나타냅니다.

- **PREDICATES WHEN = 1**은 이 함수가 DML문의 WHERE 절에서 다음과 같이 나타날 때,

  ```
  within(g1, g2) = 1
  ```

  술어가 사용자 정의 술어로 간주되고 색인 확장 *gridIndex*에 의해 정의된 색인을 사용하여 이 술어를 만족하는 행을 검색해야 함을 나타냅니다. 상수를 지정하면, DML문에서 지정된 상수는 색인 작성 명령문에 지정된 상수와 정확하게 일치해야 합니다. 이 조건은 보통 결과 유형이 1 또는 0인 BOOLEAN 표현식을 다루기 위해 제공됩니다. 다른 경우, EXPRESSION 절을 선택하는 것이 더 좋습니다.
- **FILTER USING mbrOverlap**은 더 값어치가 있는 WITHIN 술어 버전이 필터링 함수 mbrOverlap을 말합니다. 위의 예에서, mbrOverlap 함수는 입력으로 최소 경계 사각형을 취하여 신속하게 겹쳐지는 지의 여부를 판별합니다. 두 입력 형태의 최소 경계 사각형이 겹쳐지지 않으면, g1은 g2와 함께 포함되지 않습니다. 그러므로, 입력을 안전하게 버릴 수 있어서 비경제적인 WITHIN 술어의 응용프로그램을 피할 수 있습니다.
- **SEARCH BY INDEX EXTENSION** 절은 색인 확장과 검색 목표의 조합이 이 사용자 정의 술어에 사용될 수 있음을 나타냅니다.

예 6: 이 예는 유형 POINT의 두 매개변수인 P1 및 P2를 입력으로 사용하는 사용자 정의 술어 DISTANCE에 대한 윤곽을 보여줍니다.

```
CREATE FUNCTION distance (P1 POINT, P2 POINT)
     RETURNS INTEGER
     LANGUAGE C
     PARAMETER STYLE DB2SQL
     NOT VARIANT
     NOT FENCED
```

```
NO SQL
NO EXTERNAL ACTION
EXTERNAL NAME 'db2sefn!SDEDistances'
PREDICATES
WHEN > EXPRESSION AS distExpr
SEARCH BY INDEX EXTENSION gridIndex
WHEN KEY(P1) USE distanceGrRule(P2, distExpr)
WHEN KEY(P2) USE distanceGrRule(P1, distExpr)
```

DISTANCE 함수의 설명은 사용자 정의 함수의 설명과 유사하지만, 다음의 추가 사항은 이 함수가 사용자 정의 술어에서 사용될 때 그 술어가 사용자 정의 술어임을 나타냅니다.

- **PREDICATES WHEN > EXPRESSION AS distExpr**은 또 다른 유효한 술어 스펙입니다. WHEN 절에서 표현식이 지정된 경우, 그 표현식의 결과 유형은 술어가 DML문에서 사용자 정의 술어인지를 판별하는 데 사용됩니다. 예를 들면,

  ```
  SELECT T1.C1
       FROM T1, T2
       WHERE distance (T1.P1, T2.P1) > T2.C2
  ```

  술어 스펙 distance는 입력으로 두 개의 매개변수를 취하여 결과를 유형이 INTEGER인 T2.C2와 비교합니다. 오른쪽 표현식의 데이터 유형만 문제가 되므로(특정 상수를 사용하는 것과는 반대로), 비교 값으로 와일드카드를 지정하기 위해 CREATE FUNCTION DDL에서 EXPRESSION 절을 선택하는 것이 좋습니다.

  또한, 다음도 유효한 사용자 정의 술어입니다.

  ```
  SELECT T1.C1
       FROM T1, T2
       WHERE distance(T1.P1, T2.P1) > distance (T1.P2, T2.P2)
  ```

  현재, 오른쪽만 표현식으로 처리되는 제한사항이 있습니다. 왼쪽의 용어는 사용자 술어에 대한 사용자 정의 함수입니다.

- **SEARCH BY INDEX EXTENSION** 절은 색인 확장과 검색 목표의 조합이 이 사용자 정의 술어에 사용될 수 있음을 나타냅니다. distance 함수의 경우, distExpr로 식별되는 표현식도 range-producer 함수(색인 확장의 부분으로

정의된)에 전달되는 검색 인수 중 하나입니다. 표현식 식별자는 인수로 range-producer 함수에 전달되도록 표현식에 대한 이름을 정의하기 위해 사용됩니다.

CREATE FUNCTION(외부 테이블)

이 명령문은 응용프로그램 서버(AS)에 사용자 정의 외부 테이블 함수를 등록하는 데 사용됩니다.

*table function*은 SELECT의 FROM절에서 사용할 수 있으며, 한 번에 한 행씩 리턴하여 SELECT로 테이블을 리턴합니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 함수의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 함수의 스키마 이름이 존재할 경우, 스키마에 대한 CREATEIN 특권

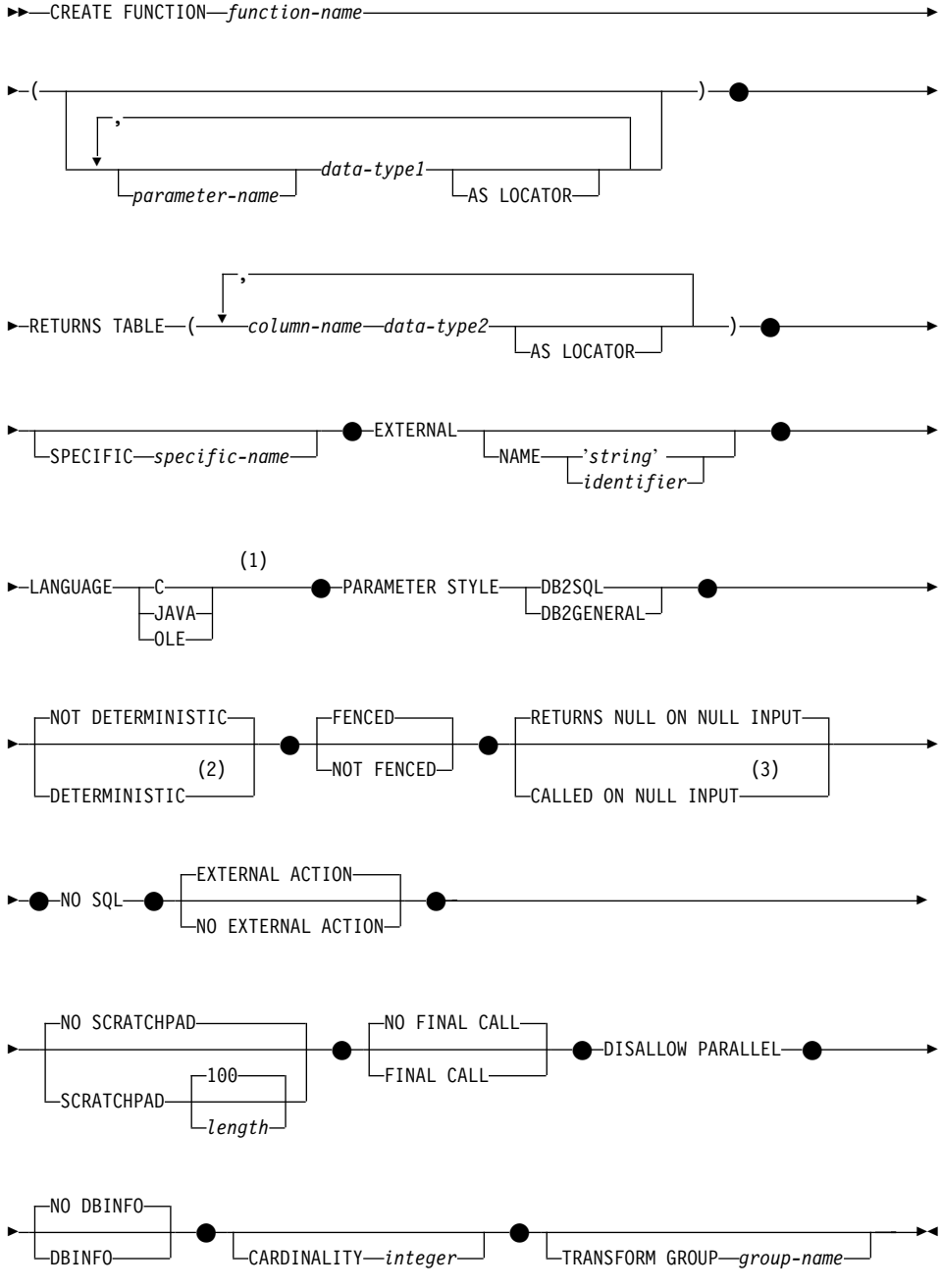
비분리 함수를 작성하려면, 명령문의 권한 부여 ID가 보유한 특권으로 다음 중 적어도 하나가 포함되어야 합니다.

- 데이터베이스에서의 CREATE_NOT_FENCED 권한
- SYSADM 또는 DBADM 권한

범위가 정해진 함수를 작성하려면, 추가 권한이나 특권이 필요하지 않습니다.

권한 부여 ID가 작업을 수행할 권한이 부족한 경우, 오류(SQLSTATE 42502)가 발생합니다.

구문


CREATE FUNCTION function-name
(
,
parameter-name
data-type1
AS LOCATOR
)
RETURNS TABLE
(
,
column-name
data-type2
AS LOCATOR
)
SPECIFIC
specific-name
EXTERNAL
NAME
'string'
identifier
LANGUAGE
C
JAVA
OLE
(1)
PARAMETER STYLE
DB2SQL
DB2GENERAL
NOT DETERMINISTIC
DETERMINISTIC
(2)
FENCED
NOT FENCED
RETURNS NULL ON NULL INPUT
CALLED ON NULL INPUT
(3)
NO SQL
EXTERNAL ACTION
NO EXTERNAL ACTION
NO SCRATCHPAD
SCRATCHPAD
100
length
NO FINAL CALL
FINAL CALL
DISALLOW PARALLEL
NO DBINFO
DBINFO
CARDINALITY
integer
TRANSFORM GROUP
group-name

주:

1 LANGUAGE OLE DB 외부 테이블 함수 작성에 대해서는 704 페이지의 『CREATE FUNCTION(OLE DB 외부 테이블)』에서 자세한 내용을 참조하십시오. LANGUAGE SQL 테이블 함수 작성에 대해서는 726 페이지의 『CREATE FUNCTION (SQL 스칼라, 테이블 또는 행)』에서 자세한 내용을 참조하십시오.

2 DETERMINISTIC 대신 NOT VARIANT를 지정할 수 있으며, NOT DETERMINISTIC 대신 VARIANT를 지정할 수 있습니다.

3 CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있으며, RETURNS NULL ON NULL INPUT 대신 NOT NULL CALL을 지정할 수 있습니다.

설명

함수 이름

정의되는 함수를 명명합니다. 이 이름은 함수를 지시하는 규정화되거나 규정화되지 않는 이름입니다. *함수 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18인)입니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다. 규정된 이름은 첫번째 매개변수가 구조화 유형일 경우 그 첫번째 매개변수의 데이터 유형과 같으면 안 됩니다.

매개변수 개수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고), 내재적 또는 명시적 규정자를 포함하여, 이름은 카탈로그에 설명된 함수를 식별하지 않아야 합니다(SQLSTATE 42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정화되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 이루어진 이름이 지정된 경우, *스키마 이름*은 “SYS”로 시작될 수 없습니다(SQLSTATE 42939).

술어에서 키워드로 사용되는 이름 중 일부는 시스템에서 사용할 수 있도록 예약되어 있어서, *함수 이름*으로 사용할 수 없습니다(SQLSTATE 42939).

SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 등이 이러한 이름이며, 217 페이지의 『기본 술어』에 기술된 비교 연산자도 여기에 포함됩니다.

함수 시그니처에 약간의 차이가 있으면, 둘 이상의 함수에 동일한 이름을 사용할 수 있습니다. 이것에 대한 제한이 없다고 하여도, 외부 사용자 정의 테이블 함수는 내장 함수와 이름이 같으면 안 됩니다.

매개변수 이름

이 함수에서 다른 모든 매개변수의 이름과 구별되는, 매개변수에 대한 선택적 이름을 지정합니다.

(데이터 유형1,...)

함수의 입력 매개변수 개수를 식별하고, 각 매개변수의 데이터 유형을 지정합니다. 함수가 받기를 기대하는 각 매개변수에 대해 목록에서 한 항목만 지정할 수 있습니다. 매개변수 개수는 90개를 초과할 수 없습니다. 이 한계를 초과하면, 오류(SQLSTATE 54023)가 발생합니다.

매개변수가 없는 함수를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예:

```
CREATE FUNCTION WOOFER() ...
```

스키마 내에서 동일하게 명명된 두 함수는 모든 해당되는 매개변수에 대해 완전히 같은 유형을 가질 수 없습니다. 길이, 정밀도 및 스케일은 이 유형의 비교에서 고려되지 않습니다. 그러므로, CHAR(8) 및 CHAR(35)는 같은 유형으로 간주되고, DECIMAL(11,2) 및 DECIMAL (4,3)도 마찬가지입니다. DECIMAL 및 NUMERIC과 같이 이러한 목적을 위해 동일한 유형으로 처리되도록 하는 유형들이 더 있습니다. 시그니처가 중복되면 SQL 오류(SQLSTATE 42723)가 발생합니다.

예를 들어, 다음 명령문이 제공되면,

```
CREATE FUNCTION PART (INT, CHAR(15)) ...
CREATE FUNCTION PART (INTEGER, CHAR(40)) ...


CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

중복되는 함수로 간주되어 두번째 및 네번째 명령문이 실패합니다.

데이터 유형1

매개변수의 데이터 유형을 지정합니다.

- CREATE TABLE문의 *데이터 유형* 정의에 지정될 수 있고 함수를 작성하는 데 사용되는 언어에서 해당사항을 갖는 SQL 데이터 유형 스펙과 약어를 지정할 수 있습니다. 사용자 정의 함수와 관련하여 SQL 데이터 유형과 호스트 언어 데이터 유형 사이의 맵핑에 관한 자세한 사항은 *응용프로그램 개발 안내서*의 해당 언어 부분을 참조하십시오.
- DECIMAL (및 NUMERIC)은 LANGUAGE C와 OLE에 대해 유효하지 않습니다(SQLSTATE 42815). DECIMAL의 대체 사용 방안에 대해서는 *응용프로그램 개발 안내서*에서 참조하십시오.
- REF(*유형 이름*)은 매개변수의 데이터 유형으로 지정할 수 있습니다. 그러나, 그러한 매개변수는 범위가 지정되지 않아야 합니다(SQLSTATE 42997).
- 연관되는 변환 그룹에 적절한 변환 함수가 존재할 경우, 구조화 유형을 지정할 수도 있습니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 LOB 위치 지정자가 실제 값 대신 UDF에 전달됨을 나타냅니다. 이로써 UDF로 전달되는 바이트 수가 상당히 절약되며, 특히 값의 일부 몇 바이트만이 UDF와 관련있는 경우 성능에도 도움이 됩니다. UDF에서 LOB 위치 지정자를 사용하는 방법은 *응용프로그램 개발 안내서*에 설명되어 있습니다.

다음은 매개변수 정의에서 AS LOCATOR절을 사용하는 예입니다.

```
CREATE FUNCTION foo ( CLOB(10M) AS LOCATOR, IMAGE AS LOCATOR)
                    ...
```

여기서는 IMAGE가 LOB 유형 중 하나에 기초한 구별 유형입니다.

인수 이송 목적의 경우, AS LOCATOR절이 아무 효과를 갖지 않습니다. 예에서, 유형은 각각 CLOB와 IMAGE로 간주되는 데, 이는 CHAR 또는 VARCHAR 인수를 첫번째 인수로 함수에 전달할 수 있

음을 의미합니다. 마찬가지로, AS LOCATOR는 함수 시그니처에 아무 영향을 미치지 않으며 이는 (a) "함수 차수"라는 프로세스에 의해 DML에서 참조될 경우와 (b) COMMENT ON이나 DROP과 같은 DDL에서 참조될 경우 함수를 서로 결합하는 데 사용됩니다. 사실상, 이 절은 COMMENT ON이나 DROP에서 다른 의미없이 사용되기도 하고 사용되지 않기도 합니다.

LOB 또는 LOB에 기초한 구별 유형 이외의 유형에 대해 AS LOCATOR가 지정되는 경우 오류(SQLSTATE 42601)가 발생합니다.

함수가 분리(fenced)된 경우, AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

RETURNS TABLE

이 함수의 결과는 테이블입니다. 이 키워드에 후속하는 괄호는 테이블의 컬럼의 이름과 유형의 목록을 한정하여, 추가 스펙(예: 제한사항)이 없는 단순 CREATE TABLE문의 양식과 유사합니다. 255개를 초과하는 컬럼은 허용되지 않습니다(SQLSTATE 54011).

컬럼 이름

이 컬럼의 이름을 지정합니다. 이름을 규정화할 수 없으며 테이블의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다.

데이터 유형2

컬럼의 데이터 유형을 지정하며, 구조화 유형을 제외하고 특정 언어로 작성된 UDF의 매개변수에 대해 지원되는 데이터 유형이면 됩니다(SQLSTATE 42997).

AS LOCATOR

*데이터 유형2*가 LOB 유형이거나 LOB 유형에 기초한 구별 유형일 경우, 이 옵션을 사용하면 함수는 결과 테이블에서 인스턴스화된 LOB 값에 대한 위치 지정자를 리턴함을 나타냅니다.

이 절에 사용할 수 있는 유효한 유형은 658페이지에 설명되어 있습니다.

SPECIFIC *특정 이름*

정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은

이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. *특정 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18인)입니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다. 암시적 또는 명시적 규정자를 포함하는 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스를 식별하지 않아야 합니다. 그렇지 않으면 오류(SQLSTATE 42710)가 발생합니다.

*고유 이름*은 기존의 *함수 이름*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *함수 이름*에 대해 사용되었던 규정자가 사용됩니다. 규정자가 지정되면, *함수 이름*의 명시적 또는 암시적 규정자와 같아야 합니다. 그렇지 않으면 오류(SQLSTATE 42882)가 발생합니다.

*특정 이름*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

EXTERNAL

이 절은 CREATE FUNCTION문이 내부 프로그래밍 언어로 작성된 코드를 기초로 문서화된 연계 규칙과 인터페이스에 따라 새 함수를 등록하는 데 사용됩니다.

NAME절이 지정되지 않을 경우, "NAME *함수 이름*"이 사용됩니다.

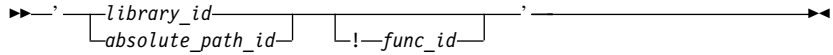
NAME *'문자열'*

이 절은 정의되는 함수를 구현하는 사용자 작성 코드를 식별합니다.

'문자열' 옵션은 최대 254 문자로 된 문자열 상수입니다. 문자열에 사용되는 형식은 지정되는 LANGUAGE에 따라 다릅니다.

- LANGUAGE C의 경우:

  지정된 *string*은 라이브러리 이름과 라이브러리 내의 함수입니다. 이것은 데이터베이스 관리자가 작성하여 사용자 정의 함수를 실행할 때 호출합니 다. CREATE FUNCTION문이 수행될 때 라이브러리(그리고 그 라이브러리 내의 함수)는 존재할 필요가 없습니다. 그러나, 함수가 SQL 문에서 사용될 때, 라이브러리와 그 라이브러리 내의 함수가 존재해야 하고 데이터베이스 서버 머신에서 접근할 수 있어야 합니다.



작은 따옴표 내에서 공백은 허용되지 않습니다.

library_id

함수를 포함하는 라이브러리 이름을 식별합니다. 데이터베이스 관리 프로그램은 .../sqllib/함수 디렉토리(UNIX용 시스템) 또는 ...*instance_name*\함수 디렉토리(OS/2, 그리고 DB2INSTPROF 레지스트리 변수에 지정된 Windows 32 비트 운영 체제)에서 그 라이브러리를 찾습니다. 이곳은 데이터베이스 관리 프로그램을 수행하기 위해 사용되는 제어하는 sqllib 디렉토리를 데이터베이스 관리 프로그램이 찾는 곳입니다. 예를 들어, UNIX 기본 시스템의 제어 sqllib 디렉토리는 /u/$DB2INSTANCE/sqllib입니다.

'myfunc'이 UNIX 기본 시스템의 *library_id*이면, 이로 인해 데이터베이스 관리 프로그램이 /u/production에서 수행될 경우에 라이브러리 /u/production/sqllib/function/myfunc에서 함수를 찾게 됩니다.

OS/2 및 Windows 32 비트 운영 체제의 경우, 데이터베이스 관리 프로그램은 *library_id*가 함수 디렉토리에 위치되지 않은 경우에 LIBPATH 또는 PATH에서 찾습니다.

OS/2에서, library_id는 8문자를 초과해선 안 됩니다.

absolute_path_id

함수의 전체 경로 이름을 식별합니다.

예를 들면, UNIX 기본 시스템에서 '/u/jchui/mylib/myfunc'으로 인해 데이터베이스 관리 프로그램이 myfunc 함수를 /u/jchui/mylib에서 찾게 됩니다.

OS/2 및 Windows 32 비트 운영 체제에서, 'd:\mylib\myfunc'는 데이터베이스 관리 프로그램이 d:\mylib 디렉토리로부터 myfunc.dll 파일을 로드하게 합니다.

OS/2에서 이 스펙의 마지막 부분(즉, dll의 이름)은 8문자를 초과해선 안 됩니다.

! func_id

호출될 함수의 입력점 이름을 식별합니다. !는 는 library id와 function id 사이의 분리 문자로 사용됩니다. *! func_id*를 생략하면, 데이터베이스 관리 프로그램은 라이브러리가 링크되었을 때 설정된 기본 진입점을 사용합니다.

예를 들면, UNIX 기본 시스템에서 'mymod!func8'은 데이터베이스 관리 프로그램이 라이브러리 $inst_home_dir/sqllib/function/mymod를 찾아 그 라이브러리 내의 진입점 func8을 사용하도록 지정합니다.

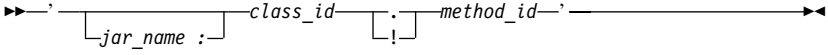
OS/2 및 Windows 32 비트 운영 체제에서, 'mymod!func8'은 데이터베이스 관리 프로그램이 mymod.dll 파일을 로드하고 동적 링크 라이브러리(DLL)에서 func8() 함수를 호출하게 합니다.

문자열이 적절히 형성되지 않으면, 오류(SQLSTATE 42878)가 발생합니다.

모든 경우에는 모든 외부 함수의 본문이 데이터베이스의 모든 파티션에서 사용 가능한 디렉토리에 있어야 합니다.

- LANGUAGE JAVA의 경우:

  지정된 *문자열*에는 선택적 jar 파일 식별자, 클래스 식별자 및 메소드 식별자가 포함됩니다. 이는 작성되는 사용자 정의 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출하는 것입니다. CREATE FUNCTION문이 수행될 때 클래스 식별자와 방법 식별자는 존재하지 않아도 됩니다. *jar_id*를 지정할 경우, CREATE FUNCTION문이 실행될 때 존재해야 합니다. 그러나, 함수가 SQL문에서 사용될 경우, 방법 식별자가 존재해야 하며, 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다.



작은 따옴표 내에서 공백은 허용되지 않습니다.

jar_name

데이터베이스에 설치될 때 jar 콜렉션에 부여된 jar 식별자를 식별

합니다. 간단한 식별자 또는 스키마 규정화된 식별자일 수 있습니다. 'myJar' 및 'mySchema.myJar'가 그 예입니다.

class_id

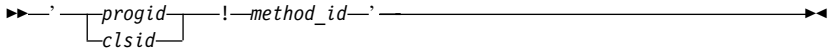
Java 오브젝트의 클래스 식별자를 나타냅니다. 클래스가 패키지의 일부일 경우, 클래스 식별자 부분에 완전한 패키지 접두어, 예를 들어, 'myPacks.UserFuncs'가 포함되어야 합니다. Java 가상 머신은 클래스를 디렉토리 '.../myPacks/UserFuncs/'에서 보게 됩니다. OS/2와 Windows 32 비트 운영 체제에서, Java 가상 머신은 디렉토리 '...\myPacks\UserFuncs\'에서 보게 됩니다.

method_id

호출할 Java 오브젝트의 방법 이름을 나타냅니다.

- LANGUAGE OLE의 경우:

  지정된 *문자열*은 OLE 프로그램가능 식별자(progid) 또는 클래스 식별자(clsid)와 방법 식별자로서, 작성중인 사용자 정의 함수를 실행하기 위해 데이터베이스 관리 프로그램이 호출한 것입니다. CREATE FUNCTION문이 수행될 때 프로그램가능 식별자, 클래스 식별자 및 방법 식별자는 존재하지 않아도 됩니다. 그러나 함수가 SQL문에서 사용될 경우, 방법 식별자가 있어야 하며 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다. 그렇지 않으면, 오류(SQLSTATE 42724)가 발생합니다.



  작은 따옴표 내에서 공백은 허용되지 않습니다.

  progid

  OLE 오브젝트의 프로그램가능 식별자를 나타냅니다.

  *progid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임 OLE API로 이송됩니다. 지정된 OLE 오브젝트는 작성 가능해야 하며 후기 바인딩(IDispatch형 바인딩이라고도 함)을 지원해야 합니다.

clsid

작성할 OLE 오브젝트의 클래스 식별자를 나타냅니다. OLE 오브젝트가 progid를 사용하여 등록되지 않은 경우 *progid*를 지정하기 위한 대안으로 사용할 수 있습니다. *clsid*은 다음과 같은 형식을 갖습니다.

{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}

여기서, 'n'은 영숫자 문자입니다. *clsid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임 OLE API로 이송됩니다.

method_id

호출할 OLE 오브젝트의 방법 이름을 나타냅니다.

NAME *식별자*

이 절은 정의되는 함수를 구현하는 사용자 작성 코드의 이름을 식별합니다. 지정된 *identifier*는 SQL 식별자입니다. SQL 식별자는 문자열에서 *library id*로 사용됩니다. 분리 식별자가 아닐 경우, 식별자는 대문자로 겹쳐집니다. 식별자에 스키마 이름이 규정된 경우, 스키마 이름 부분은 무시됩니다. 이 형식의 이름은 LANGUAGE C에서만 사용할 수 있습니다.

LANGUAGE

이 필수 절은 사용자 정의 함수의 본체가 작성되는 언어 인터페이스 규약을 지정하는 데 사용됩니다.

C 즉, 데이터베이스 관리 프로그램은 C 함수처럼 사용자 정의 함수를 호출합니다. 사용자 정의 함수는 표준 ANSI C 프로토타입에 의해 정의된 대로 C 언어 호출 규칙과 연계 규칙에 따라야 합니다.

JAVA 이것은 데이터베이스 관리 프로그램이 Java 클래스의 한 방법으로 사용자 정의 함수를 호출하는 것을 의미합니다.

OLE 데이터베이스 관리 프로그램은 OLE 자동화 오브젝트에 의해 노출된 방법인 것처럼 사용자 정의 함수를 호출합니다. 사용자 정의 함수는 *OLE Automation Programmer's Reference*에 기술된 것처럼 OLE 자동화 데이터 유형 및 호출 메카니즘과 일치해야 합니다.

LANGUAGE OLE는 Windows 32 비트 운영 체제용 DB2에 저장된 사용자 정의 함수에 대해서만 지원됩니다

LANGUAGE OLE DB 외부 테이블 함수 작성에 대해서는 704 페이지의 『CREATE FUNCTION(OLE DB 외부 테이블)』에서 참조하십시오.

PARAMETER STYLE

이 절은 함수로부터 값을 리턴하고 매개변수를 전달하기 위한 규약을 지정하는 데 사용됩니다.

DB2SQL

이 필수 절은 C 언어 호출 및 연결 규약에 부합하는 외부 함수로부터 값을 리턴하고 매개변수를 전달하는 데 사용되는 규약을 지정하는 데 사용됩니다. LANGUAGE C 또는 LANGUAGE OLE를 사용할 때 이를 지정해야 합니다.

DB2GENERAL

이 절은 Java 클래스에 방법으로 정의된 외부 함수로부터 값을 리턴하고 매개변수를 전달하는 데 사용할 규약을 지정하는 데 사용됩니다. 이것은 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

값 DB2GENRL는 DB2GENERAL의 동의어로 사용할 수 있습니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**

이 선택적 절은 함수가 항상 주어진 값에 대해 동일한 결과를 리턴하는지(DETERMINISTIC) 또는 함수가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 함수는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC를 지정하여 방지할 수 있습니다. NOT DETERMINISTIC 테이블 함수의 한 예로는 파일과 같은 소스의 데이터를 검색하는 함수를 들 수 있습니다.

FENCED 또는 **NOT FENCED**

이 절은 함수가 데이터베이스 관리 프로그램 운영 환경의 프로세스나 주소 공간에서 수행하기에 "안전한"지(NOT FENCED) 않은지(FENCED)를 지정합니다.

함수가 FENCED로 등록되면, 데이터베이스 관리 프로그램은 함수에서 액세스를 못하도록 내부 자원(예를 들어, 데이터 버퍼)을 분리시킵니다. 대부분의 함

수는 FENCED나 NOT FENCED로 수행되는 옵션을 갖고 있습니다. 일반적으로, FENCED로 수행되는 함수는 NOT FENCED로 수행되는 것과 유사하게 수행되지 않습니다.

경고: 적절히 코딩, 검토 및 테스트되지 않은 함수에 대해 NOT FENCED를 사용하면 DB2의 무결성을 보완할 수 있습니다. DB2는 발생할 수도 있는 많은 공통 유형의 실수에 대한 예방책을 취하나, NOT FENCED 사용자 정의 함수가 사용될 때 완전한 무결성을 보증할 수 없습니다.

FENCED를 사용하면 데이터베이스 무결성에 대해 더 나은 보호 수준을 제공하지만, 적절하게 코딩, 검토 및 테스트되지 않은 FENCED UDF는 DB2 장애를 야기할 수도 있습니다.

대부분의 사용자 정의 함수는 FENCED나 NOT FENCED로 수행될 수 있어야 합니다. LANGUAGE OLE에서는 함수에 대해 FENCED만을 지정할 수 있습니다(SQLSTATE 42613).

FENCED에서 NOT FENCED로 변경하려면, 함수는 다시 등록되어야 합니다(먼저 그 함수를 삭제한 후 다시 작성). 사용자 정의 함수를 NOT FENCED로 등록하려면 SYSADM 권한, DBADM 권한 또는 특수 권한(CREATE_NOT_FENCED)이 필요합니다.

함수가 분리(fenced)된 경우, AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

RETURNS NULL ON NULL INPUT 또는 CALLED ON NULL INPUT

이 선택적 절은 널(NULL) 값인 인수가 있을 경우 외부 함수로의 호출을 방지하기 위해 사용됩니다. 사용자 정의 함수(UDF)가 아무 매개변수도 가지지 않는 것으로 정의되면, 물론 이러한 널(NULL) 인수 상태가 발생할 수 없으며 이 스펙의 코딩 방법이 문제가 되지 않습니다.

RETURNS NULL ON NULL INPUT이 지정되고 테이블 함수 OPEN 시기에 함수 인수 중 어느 하나가 널(NULL)일 경우, 사용자 정의 함수는 호출되지 않습니다. 시도된 테이블 함수 스캔의 결과는 빈 테이블(행이 없는 테이블)입니다.

CALLED ON NULL INPUT이 지정될 경우, 어느 인수가 널(NULL)인지에 관계 없이 사용자 정의 함수가 호출됩니다. 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 그러나, 널(NULL) 값 인수 값에 대한 테스트 책임은 UDF에 있습니다.

NULL CALL이 역방향의 계열 호환성을 위해 CALLED ON NULL INPUT에 대한 동의어로서 사용될 수도 있습니다. 마찬가지로, NOT NULL CALL이 RETURNS NULL ON NULL INPUT에 대한 동의어로서 사용될 수도 있습니다.

NO SQL

이 필수 절은 함수가 SQL문을 실행할 수 없음을 나타냅니다. 이를 수행하면 런타임 오류(SQLSTATE 38502)가 발생합니다.

NO EXTERNAL ACTION 또는 **EXTERNAL ACTION**

이 선택적 절은 데이터베이스 관리 프로그램에 의해 관리되지 않는 오브젝트의 상태를 변경하는 일부 조치를 함수에서 사용하고 있는 지를 지정합니다. 외부 영향이 없는 함수를 가정하는 최적화는 EXTERNAL ACTION을 지정하여 금지됩니다. 예를 들어, 메시지 전송, 벨 울림 또는 파일에 레코드 기록.

NO SCRATCHPAD 또는 **SCRATCHPAD** *길이*

이 선택적 절은 외부 함수에 대해 스크래치패드가 제공되는지의 여부를 지정할 때 사용할 수 있습니다(사용자 정의 함수는 재입력이 가능하도록 적극 권장되므로, 스크래치패드는 함수가 한 호출에서 다음 호출로의 "상태를 저장" 하기 위한 수단을 제공합니다.).

SCRATCHPAD가 지정되면, 사용자 정의 함수를 처음 호출할 때 외부 함수에서 사용할 스크래치패드에 메모리가 할당됩니다. 이 스크래치패드는 다음 특성을 갖습니다.

- *길이*를 지정할 경우, 이것은 스크래치 패드의 크기를 바이트 단위로 설정합니다. 이 값은 1 - 32 767 사이여야 합니다(SQLSTATE 42820). 기본 값은 100입니다.
- 모든 X'00'에 대해 초기화됩니다.
- 범위는 SQL문입니다. SQL문에서 외부 함수에 대해 참조당 하나의 스크래치패드가 있습니다. 그래서, 다음 명령문의 UDFX 함수가 SCRATCHPAD 키워드를 사용하여 정의되는 경우, 두 개의 스크래치패드가 지정됩니다.

```
SELECT A.C1, B.C2
   FROM TABLE (UDFX(:hv1)) AS A,
        TABLE (UDFX(:hv1)) AS B
       WHERE ...
```

- 지속적입니다. 이는 명령문의 실행 초기에 초기화되며, 호출간에 스크래치패드의 상태를 유지하기 위해 외부 테이블 함수에 의해 사용될 수 있습니다. UDF에 대해서도 FINAL CALL 키워드가 지정되면, 스크래치패드는 절대 DB2에 의해 변경되지 않으며 스크래치패드에 할당된 모든 자원은 특수 FINAL 호출이 발행되면 해제됩니다.

  NO FINAL CALL이 지정되었거나 기본값이면, 외부 테이블 함수는 CLOSE 호출시 이러한 자원을 지우고 DB2는 OPEN 호출 시마다 스크래치패드를 다시 초기화합니다. 테이블 함수가 부속 조회나 조인(join)에 사용될 경우에는 FINAL CALL 또는 NO FINAL CALL 및 스크래치 패드의 관련 동작을 판별하는 것이 매우 중요한데, 이유는 이 때가 명령문 실행중에 여러 개의 OPEN 호출이 발생할 수 있는 때이기 때문입니다.

- 외부 함수가 획득할 수 있는 시스템 자원(예: 메모리)에 대해 중심점으로 사용됩니다. 함수는 첫번째 호출에서 메모리를 획득하고, 그 주소를 스크래치패드에 보관한 후 다음 호출에서 이를 참조할 수 있습니다.

  (위에서 간단히 설명했듯이, FINAL CALL/NO FINAL CALL 키워드는 스크래치패드의 재 초기화를 제어하는 데 사용되며, 외부 테이블 함수가 스크래치패드에 할당된 자원을 해제하는 시기도 말해줍니다.)

SCRATCHPAD가 지정되면, 사용자 정의 함수를 호출할 때마다 추가 인수가 스크래치패드가 주소지정된 외부 함수로 전달됩니다.

NO SCRATCHPAD가 지정되면, 어떤 스크래치패드도 외부 함수에 할당되거나 전달되지 않습니다.

NO FINAL CALL 또는 FINAL CALL

이 선택적 절은 외부 함수에 대해 마지막 호출(및 별도의 첫번째 호출)이 수행되는지 여부를 지정합니다. 또한 언제 스크래치패드가 다시 초기화되는지도 제어합니다. NO FINAL CALL이 지정되면, DB2는 테이블 함수에 대해 세 가지 유형의 호출(열기, 페치 및 닫기)만을 발행할 수 있습니다. 단, FINAL

CALL이 지정되면, 테이블 함수에 대해 열기, 페치 및 닫기 외에도 첫번째 호출(first call) 및 마지막 호출(final call)을 할 수 있습니다.

외부 테이블 함수의 경우, 어떤 옵션을 선택하는지에 관계없이 호출 유형 인수는 항상 존재합니다. 이 인수 및 값에 대한 *응용프로그램 개발 안내서*에서 좀더 자세한 정보를 참조하십시오.

오류 발생시 이들 호출의 테이블 UDF 처리 설명은 *응용프로그램 개발 안내서*에 포함됩니다.

DISALLOW PARALLEL

이 절은 함수에 대한 단일 참조에 대해 함수의 호출이 병렬화될 수 없음을 지정합니다. 테이블 함수는 항상 단일 파티션에서 수행됩니다.

NO DBINFO 또는 **DBINFO**

이 선택적 절은 DB2가 확인한 특정 정보가 추가 호출 인수로 UDF에 전달되는지(DBINFO) 않은지(NO DBINFO)의 여부를 지정합니다. NO DBINFO가 기본값입니다. DBINFO는 LANGUAGE OLE에서는(SQLSTATE 42613) 지원하지 않습니다.

DBINFO가 지정되는 경우, 다음 정보가 들어 있는 UDF로 구조가 전달됩니다.

- 데이터베이스 이름 - 현재 연결된 데이터베이스의 이름.
- 응용프로그램 ID - 데이터베이스로의 각 연결에 형성되는 고유한 응용프로그램 ID.
- 응용프로그램 권한 부여 ID - 이 UDF와 응용프로그램 사이의 중첩 UDF에 관계없이 응용프로그램 런타임 권한 부여 ID.
- 코드 페이지 - 데이터베이스 코드 페이지를 식별합니다.
- 스키마 이름 - 외부 테이블 함수에는 적용되지 않음
- 테이블 이름 - 외부 테이블 함수에는 적용되지 않습니다.
- 컬럼 이름 - 외부 테이블 함수에는 적용되지 않습니다.
- 데이터베이스 버전/릴리스 - UDF를 호출한 데이터베이스 서버의 버전, 릴리스 및 개정 레벨을 나타냅니다.
- 플랫폼 - 서버의 플랫폼 유형을 포함합니다.

- 테이블 함수 결과 컬럼 수 - 함수를 참조하는 특정 명령문에 요구되는 테이블 함수 결과 컬럼의 수의 배열. 테이블 함수에 대해서만 제공되는 경우, UDF는 모든 컬럼 값을 리턴하는 대신 필수 컬럼 값만을 리턴하여 최적화할 수 있습니다.

구조와 이 구조가 UDF에 전달되는 방식에 대한 *응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.

CARDINALITY *정수*

이 선택적 절은 최적화를 위해 함수가 리턴하는 예상 행의 수를 제공합니다. *integer*에 대한 유효한 값의 범위는 0에서 2 147 483 647까지입니다.

테이블 함수에 대해 CARDINALITY절이 지정되지 않은 경우, DB2는 유한 값을 기본값으로 가정하는 데, 이 값은 RUNSTATS가 통계를 수집하지 않은 테이블에 추정되는 값과 동일합니다.

경고: 함수가 무한 기본 행수를 가지고 있는 경우, 즉, 요청될 때마다 행을 리턴하고 절대 "테이블의 끝" 조건을 리턴하지 않는 경우, 정확하게 기능하려면 "테이블의 끝" 조건을 필요로 하는 조회는 무한하게 되어 인터럽트됩니다. GROUP BY와 ORDER BY가 연관된 조회가 이러한 조회에 해당합니다. 이러한 UDF는 작성하지 않는 것이 좋습니다.

TRANSFORM GROUP *그룹 이름*

함수를 호출할 때 사용자가 정의하는 구조화 유형변환에 대해 사용되는 변환 그룹을 나타냅니다. 변환은 함수 정의에 매개변수로 사용자 정의 구조화 유형을 포함할 경우에 필요합니다. 이 절을 지정하지 않으면, 기본 그룹 이름인 DB2_FUNCTION이 사용됩니다. 지정된 (또는 기본값인) *그룹 이름*이 참조된 구조화 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42741). 필수 FROM SQL 변환 함수가 제공된 그룹 이름과 구조화 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42744).

주

- 사용자 정의 함수의 매개변수에 대한 데이터 유형을 선택할 때, 그 입력 값에 영향을 주게 될 승격 규칙을 고려하십시오(104 페이지의 『데이터 유형의 승격』 참조). 예를 들어, 입력 값으로 사용된 상수가 예상한 것과 다른 내장 데이터

유형을 가질 수도 있고, 좀더 심각하게는 예상한 데이터 유형으로 승격되지 않을 수 있습니다. 승격 규칙에 따라, 매개변수에 다음의 데이터 유형을 사용하는 것이 좋습니다.

  - SMALLINT 대신 INTEGER
  - REAL 대신 DOUBLE
  - CHAR 대신 VARCHAR
  - GRAPHIC 대신 VARGRAPHIC

- 여러 플랫폼에서의 UDF 이식성을 위해 다음과 같은 데이터 유형을 사용해서는 안 됩니다.
  - FLOAT- DOUBLE 또는 REAL을 대신 사용
  - NUMERIC- DECIMAL을 대신 사용
  - LONG VARCHAR- CLOB(또는 BLOB)를 대신 사용
- 외부 사용자 정의 함수의 작성, 컴파일 및 링크에 대한 *응용프로그램 개발 안내서*에서 정보를 참조하십시오.
- 아직 존재하지 않는 스키마 이름을 사용하여 함수를 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.

예

예 1: 다음은 텍스트 관리 시스템에 알려진 도큐멘트에 대해 단일 도큐멘트 식별자 컬럼으로 이루어진 행을 리턴하는 데 기록된 테이블 함수를 등록합니다. 첫번째 매개변수는 주어진 주제 영역과 일치를 이루고 두 번째 매개변수는 주어진 문자열을 포함합니다.

```
단일 세션의 문맥에서 UDF는 항상 같은 테이블을 리턴합니다. 그래서
DETERMINISTIC로 정의됩니다. DOCMATCH에서 출력을 정의한 RETURNS
절에 유의하십시오. FINAL CALL는 각 테이블 함수에 대해 지정되어야 합니다.
또한, DISALLOW PARALLEL 키워드는 평행하게 조작할 수 없는 테이블 함수
로 추가됩니다. DOCMATCH에 대해 출력의 크기가 변수일지라도 CARDINALITY
20가 나타내는 값이며 DB2 최적화 알고리즘을 돕기 위해 지정됩니다.
```

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
    RETURNS TABLE (DOC_ID CHAR(16))
    EXTERNAL NAME '/common/docfuncs/rajiv/udfmatch'
    LANGUAGE C
    PARAMETER STYLE DB2SQL
       NO SQL
       DETERMINISTIC
       NO EXTERNAL ACTION
    NOT FENCED
    SCRATCHPAD
    FINAL CALL
      DISALLOW PARALLEL
    CARDINALITY 20
```

예 2: 다음은 Microsoft Exchange에서 텍스트의 부분 메시지와 메시지 헤더 정보를 검색하는 데 사용된 OLE 테이블 함수를 등록합니다. 테이블 함수를 구현하는 코드의 예에 대해서는 *응용프로그램 개발 안내서*에서 참조하십시오.

```
CREATE FUNCTION MAIL()
    RETURNS TABLE (TIMERECIEVED DATE,
                   SUBJECT VARCHAR(15),
                   SIZE INTEGER,
                   TEXT VARCHAR(30))
    EXTERNAL NAME 'tfmail.header!list'
    LANGUAGE OLE
    PARAMETER STYLE DB2SQL
    NOT DETERMINISTIC
       FENCED
       CALLED ON NULL INPUT
    SCRATCHPAD
    FINAL CALL
       NO SQL
    EXTERNAL ACTION
      DISALLOW PARALLEL
```

CREATE FUNCTION(OLE DB 외부 테이블)

이 명령문은 OLE DB 공급자로부터의 데이터를 액세스할 수 있게 사용자 정의 OLE DB 외부 테이블 함수를 등록하는 데 사용됩니다.

*테이블 함수*는 SELECT의 FROM절에서 사용될 수도 있습니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

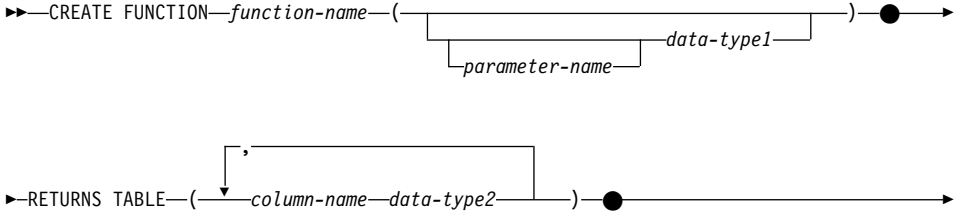
권한 부여

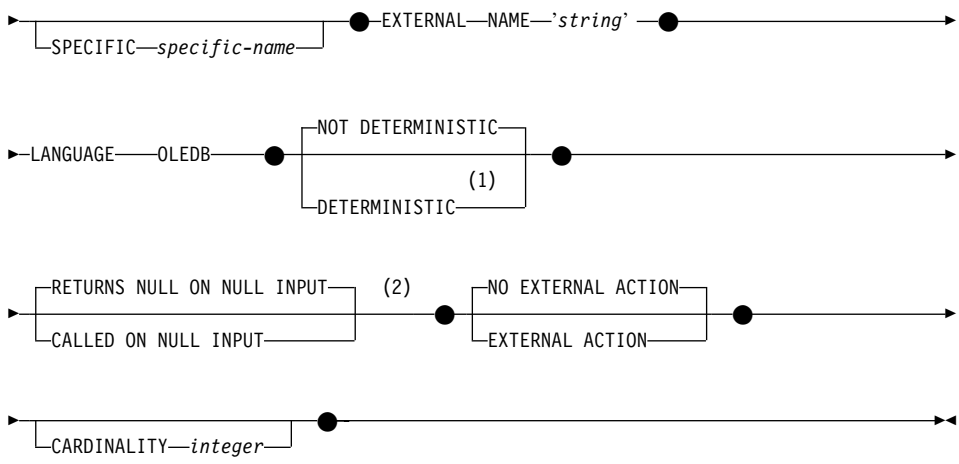
명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 함수의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 함수의 스키마 이름이 존재할 경우, 스키마에 대한 CREATEIN 특권

권한 부여 ID가 작업을 수행할 권한이 부족한 경우, 오류(SQLSTATE 42502)가 발생합니다.

구문





주:

1 DETERMINISTIC 대신 NOT VARIANT를 지정할 수 있으며, NOT DETERMINISTIC 대신 VARIANT를 지정할 수 있습니다.

2 CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있으며, RETURNS NULL ON NULL INPUT 대신 NOT NULL CALL을 지정할 수 있습니다.

설명

함수 이름

정의되는 함수를 명명합니다. 이 이름은 함수를 지시하는 규정화되거나 규정화되지 않는 이름입니다. *함수 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18인)입니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다.

매개변수 개수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고), 내재적 또는 명시적 규정자를 포함하여, 이름은 카탈로그에 설명된 함수를 식별하지 않아야 합니다(SQLSTATE

42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정화되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 이루어진 이름이 지정된 경우, *스키마 이름*은 "SYS"로 시작될 수 없습니다(SQLSTATE 42939).

술어에서 키워드로 사용되는 이름 중 일부는 시스템에서 사용할 수 있도록 예약되어 있어서, *함수 이름*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH와 같은 이름들이 있고, 217 페이지의 『기본 술어』에 기술된 비교 연산자도 여기에 포함됩니다.

함수 시그니처에 약간의 차이가 있으면, 둘 이상의 함수에 동일한 이름을 사용할 수 있습니다. 이것에 대한 제한이 없다고 하여도, 외부 사용자 정의 테이블 함수는 내장 함수와 이름이 같으면 안 됩니다.

매개변수 이름

매개변수에 대한 선택적 이름을 지정합니다.

데이터 유형1

함수의 입력 매개변수를 식별하며, 매개변수의 데이터 유형을 지정합니다. 아무 입력 매개변수도 지정되지 않으면, 아마도 조회 최적화를 통해 부속 집합된 외부 소스로부터 데이터가 검색됩니다. 입력 매개변수는 임의의 문자 또는 그래픽 문자열 데이터 유형일 수 있으며 명령 텍스트를 OLE DB 공급자에게 전달합니다.

매개변수가 없는 함수를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예:

```
CREATE FUNCTION WOOFER() ...
```

스키마 내에서 동일하게 명명된 두 함수는 모든 해당되는 매개변수에 대해 완전히 같은 유형을 가질 수 없습니다. 길이는 이 유형 비교에서 고려되지 않습니다. 그러므로 CHAR(8)과 CHAR(35)가 같은 유형으로 간주됩니다. 시그니처가 중복되면 SQL 오류(SQLSTATE 42723)가 발생합니다.

RETURNS TABLE

이 함수의 결과는 테이블입니다. 이 키워드에 후속하는 괄호는 테이블의 컬럼

의 이름과 유형의 목록을 한정하여, 추가 스펙(예: 제한사항)이 없는 단순 CREATE TABLE문의 양식과 유사합니다.

컬럼 이름

해당 rowset 컬럼 이름과 동일해야 하는 컬럼의 이름을 지정합니다. 이름을 규정화할 수 없으며 테이블의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다.

데이터 유형2

컬럼의 데이터 유형을 지정합니다(SQL 데이터 유형과 OLE DB 데이터 유형간의 맵핑에 대한 세부사항은 *응용프로그램 개발 안내서*의 해당 언어 부분을 참조하십시오.).

SPECIFIC *특정 이름*

정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. *특정 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18)입니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다. 암시적 또는 명시적 규정자를 포함하는 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스를 식별하지 않아야 합니다. 그렇지 않으면 오류(SQLSTATE 42710)가 발생합니다.

*고유 이름*은 기존의 *함수 이름*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *함수 이름*에 대해 사용되었던 규정자가 사용됩니다. 규정자가 지정되면 *함수 이름*의 명시적 또는 암시적 규정자와 같아야 합니다. 그렇지 않으면 오류(SQLSTATE 42882)가 발생합니다.

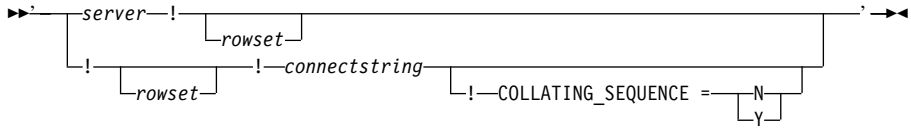
*특정 이름*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

EXTERNAL NAME *'문자열'*

이 절은 외부 테이블 및 OLE DB 공급자를 식별합니다.

'문자열' 옵션은 최대 254 문자로 된 문자열 상수입니다.

지정된 문자열은 OLE DB 공급자와의 연결 및 세션을 형성하고, rowset으로부터 데이터를 검색하는 데 사용됩니다. CREATE FUNCTION문이 수행될 때 OLE DB 공급자와 데이터 소스가 존재할 필요는 없습니다. 세부사항은 *응용프로그램 개발 안내서*에서 OLE DB 테이블 함수를 참조하십시오.



서버

795 페이지의 『CREATE SERVER』에 의해 정의된 대로 데이터 소스의 지역 이름을 식별합니다.

rowset

OLE DB 공급자에 의해 노출된 rowset(테이블)을 식별합니다. 카탈로그나 스키마 이름을 지원하는 OLE DB 공급자에 대해 완전한 테이블 이름이 제공되어야 합니다.

connectstring

데이터 소스에 연결하는 데 필요한 초기화 등록 정보의 문자열 버전. 연결 문자열의 기본 형식은 ODBC 연결 문자열에 기초합니다. 문자열은 세미콜론으로 분리되는 일련의 키워드/값 쌍을 포함합니다. 등호(=)는 각 키워드와 해당되는 값을 구분합니다. 키워드는 OLE DB 초기화 등록 정보(등록 정보 세트 DBPROPSET_DBINIT) 또는 공급자 고유 키워드의 설명입니다. 세부사항은 *응용프로그램 개발 안내서*의 해당 언어 부분을 참조하십시오.

COLLATING_SEQUENCE

데이터 소스가 DB2 Universal Database와 동일한 조합 순서를 사용하는지의 여부를 지정합니다. 795 페이지의 『CREATE SERVER』에서 자세한 내용을 참조하십시오. 유효한 값은 다음과 같습니다.

Y = 동일 조합 순서

N = 다른 조합 순서

COLLATING_SEQUENCE가 지정되지 않으면, 데이터 소스가 DB2 Universal Database와 서로 다른 조합 순서를 가지는 것으로 가정합니다.

*server*가 제공되면, *connectstring* 또는 COLLATING_SEQUENCE가 외부 이름에서 허용되지 않습니다. 이들은 서버 옵션 CONNECTSTRING 및 COLLATING_SEQUENCE로서 정의됩니다. 아무 *server*도 제공되지 않으면, *connectstring*을 제공해야 합니다. *rowset*이 제공되지 않으면, 테이블 함수에 명령 텍스트를 통해 OLE DB 공급자에 전달하는 입력 매개변수가 있어야 합니다.

LANGUAGE OLEDB

이는 데이터베이스 관리 프로그램이 OLE DB 공급자로부터 데이터를 검색하는 내장 일반 OLE DB 소비자를 배치할 것임을 의미합니다. 개발자는 아무 테이블 함수 구현도 필요로 하지 않습니다.

LANGUAGE OLEDB 테이블 함수는 어느 플랫폼에서든 작성될 수 있으나, Microsoft OLE DB에 의해 지원되는 플랫폼에서만 실행될 수 있습니다.

DETERMINISTIC 또는 NOT DETERMINISTIC

이 선택적 절은 함수가 항상 주어진 값에 대해 동일한 결과를 리턴하는지(DETERMINISTIC) 또는 함수가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 함수는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC를 지정하여 방지할 수 있습니다.

RETURNS NULL ON NULL INPUT 또는 CALLED ON NULL INPUT

이 선택적 절은 널(NULL) 값인 인수가 있을 경우 외부 함수로의 호출을 방지하기 위해 사용됩니다. 사용자 정의 함수(UDF)가 아무 매개변수도 가지지 않는 것으로 정의되면, 물론 이 널(NULL) 인수 조건이 일어날 수가 없습니다.

RETURNS NULL ON NULL INPUT이 지정되고 실행 시 함수 인수 중 어느 하나가 널(NULL)일 경우, 사용자 정의 함수는 호출되지 않고 결과는 빈 테이블(예를 들어, 행이 전혀 없는 테이블)입니다.

CALLED ON NULL INPUT이 지정될 경우, 실행 시 인수가 널(NULL)인지에 관계 없이 사용자 정의 함수가 호출됩니다. 자체 로직에 따라 공백 테이블을 리턴하거나 그렇지 않을 수 있습니다. 그러나, 널(NULL) 값 인수 값에 대한 테스트 책임은 UDF에 있습니다.

NULL CALL이 역방향 및 계열 호환성을 위해 CALLED ON NULL INPUT에 대한 동의어로서 사용될 수도 있습니다. 마찬가지로, NOT NULL CALL이 RETURNS NULL ON NULL INPUT에 대한 동의어로서 사용될 수도 있습니다.

NO EXTERNAL ACTION 또는 **EXTERNAL ACTION**

이 선택적 절은 데이터베이스 관리 프로그램에 의해 관리되지 않는 오브젝트의 상태를 변경하는 일부 조치를 함수에서 사용하고 있는 지를 지정합니다. 외부 영향이 없는 함수를 가정하는 최적화는 EXTERNAL ACTION을 지정하여 금지됩니다. 예를 들어, 메시지 전송, 벨 울림 또는 파일에 레코드 기록.

CARDINALITY *정수*

이 선택적 절은 최적화를 위해 함수가 리턴하는 예상 행의 수를 제공합니다. *integer*에 대한 유효한 값의 범위는 0에서 2 147 483 647까지입니다.

테이블 함수에 대해 CARDINALITY절이 지정되지 않은 경우, DB2는 유한 값을 기본값으로 가정하는 데, 이 값은 RUNSTATS가 통계를 수집하지 않은 테이블에 추정되는 값과 동일합니다.

경고: 함수가 무한 기본 행수를 가지고 있는 경우, 즉, 요청될 때마다 행을 리턴하고 절대 ″테이블의 끝″ 조건을 리턴하지 않는 경우, 정확하게 기능하려면 ″테이블의 끝″ 조건을 필요로 하는 조회는 무한하게 되어 인터럽트됩니다. GROUP BY와 ORDER BY가 연관된 조회가 이러한 조회에 해당합니다. 이러한 UDF는 작성하지 않는 것이 좋습니다.

주

- FENCED, FINAL CALL, SCRATCHPAD, PARAMETER STYLE DB2SQL, DISALLOW PARALLEL, NO DBINFO 및 NO SQL은 명령문에서 내재적이며 지정될 수 없습니다. 특정 설명에 대해서는 685 페이지의 『CREATE FUNCTION(외부 테이블)』에서 참조하십시오.

- 사용자 정의 함수의 매개변수에 대한 데이터 유형을 선택할 때, 그 입력 값에 영향을 주게 될 승격 규칙을 고려하십시오(104 페이지의 『데이터 유형의 승격』 참조). 예를 들어, 입력 값으로 사용된 상수가 예상한 것과 다른 내장 데이터 유형을 가질 수도 있고, 좀더 심각하게는 예상한 데이터 유형으로 승격되지 않을 수 있습니다. 승격 규칙에 따라, 매개변수에 다음의 데이터 유형을 사용하는 것이 좋습니다.
  - CHAR 대신 VARCHAR
  - GRAPHIC 대신 VARGRAPHIC
- 여러 플랫폼에서의 UDF 이식성을 위해 다음과 같은 데이터 유형을 사용해서는 안됩니다.
  - FLOAT- DOUBLE 또는 REAL을 대신 사용
  - NUMERIC- DECIMAL을 대신 사용
  - LONG VARCHAR- CLOB(또는 BLOB)를 대신 사용
- 사용자 정의 OLE DB 외부 테이블 함수에 대한 *응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.
- 아직 존재하지 않는 스키마 이름을 사용하여 함수를 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.

예

예 1: 다음은 Microsoft Access 데이터베이스로부터 정보를 검색하는 OLE DB 테이블 함수를 등록합니다. 연결 문자열이 외부 이름에서 정의됩니다.

```
CREATE FUNCTION orders ()
  RETURNS TABLE (orderid INTEGER,
                 customerid CHAR(5),
                 employeeid INTEGER,
                 orderdate TIMESTAMP,
                 requireddate TIMESTAMP,
                 shippeddate TIMESTAMP,
                 shipvia INTEGER,
                 freight dec(19,4))
  LANGUAGE OLEDB
  EXTERNAL NAME '!orders!Provider=Microsoft.Jet.OLEDB.3.51;
                          Data Source=c:\sqllib\samples\oledb\nwind.mdb
  !COLLATING_SEQUENCE=Y';
```

예 2: 다음은 Oracle 데이터베이스로부터 정보를 검색하는 OLE DB 테이블 함수를 등록합니다. 연결 문자열은 서버 정의를 통해 제공됩니다. 테이블 이름은 외부 이름에서 완전하게 규정화됩니다. 국지 사용자 john이 원격 사용자 dave로 맵핑됩니다. 다른 사용자들은 문자열을 연결하는 데 guest userid를 사용할 것입니다. 명령문에 대한 795 페이지의 『CREATE SERVER』, 949 페이지의 『CREATE WRAPPER』 및 928 페이지의 『CREATE USER MAPPING』에서 자세한 내용을 참조하십시오.

```
CREATE SERVER spirit
  WRAPPER OLEDB
  OPTIONS (CONNECTSTRING 'Provider=MSDAORA;Persist Security Info=False;
                          User ID=guest;password=pwd;Locale Identifier=1033;
                          OLE DB Services=CLIENTCURSOR;Data Source=spirit');
CREATE USER MAPPING FOR john
  SERVER spirit
  OPTIONS (REMOTE_AUTHID 'dave', REMOTE_PASSWORD 'mypwd');
CREATE FUNCTION customers ()
  RETURNS TABLE (customer_id INTEGER,
                 name  VARCHAR(20),
                 address VARCHAR(20),
                 city VARCHAR(20),
                 state VARCHAR(5),
                 zip_code INTEGER)
  LANGUAGE OLEDB
  EXTERNAL NAME 'spirit!demo.customer';
```

예 3: 다음은 MS SQL Server 7.0 데이터베이스로부터 정보를 검색하는 OLE DB 테이블 함수를 등록합니다. 연결 문자열은 외부 이름에서 제공됩니다. 테이블 함수는 명령 텍스트를 통해 OLE DB 공급자에게 전달하는 입력 매개변수를 가집니다. rowset 이름은 외부 이름에서 지정될 필요가 없습니다. 조회 예는 SQL 명령 텍스트로 전달되어 상위 세 개의 점포를 검색합니다.

```
CREATE FUNCTION favorites (varchar(600))
  RETURNS TABLE (store_id CHAR (4),
                 name  VARCHAR (41),
                 sales INTEGER)
  SPECIFIC favorites
  LANGUAGE OLEDB
  EXTERNAL NAME '!!Provider=SQLOLEDB.1;Persist Security Info=False;
                 User ID=sa;Initial Catalog=pubs;Data Source=WALTZ;
                 Locale Identifier=1033;Use Procedure for Prepare=1;
                 Auto Translate=False;Packet Size=4096;Workstation ID=WALTZ;
                 OLE DB Services=CLIENTCURSOR;';
SELECT *
  FROM TABLE (favorites
              (' select top 3 sales.stor_id as store_id, ' || '
                   stores.stor_name as name, ' || '
                   sum(sales. qty) as sales  ' || '
                from sales, stores ' || '
```

```
where sales.stor_id = stores.stor_id ' || '
group by sales.stor_id, stores.stor_name' || '
order by sum(sales.qty) desc')) as f;
```

CREATE FUNCTION (소스 또는 템플리트)

이 명령문은 다음을 수행하는 데 사용됩니다.

- 기존의 또 다른 스칼라 함수나 컬럼 함수에 기초하여 사용자 정의 함수를 응용프로그램 서버에 등록합니다.
- 함수 템플리트를 연합 서버로 지정되는 응용프로그램 서버에 등록합니다. *함수 템플리트*는 실행 가능 코드를 포함하고 있지 않는 부분적인 함수입니다. 사용자는 데이터 소스 함수에 맵핑할 목적으로 이를 작성합니다. 맵핑이 작성된 후, 사용자는 연합 서버로 제출된 조회에서 함수 템플리트를 지정할 수 있습니다. 그러한 조회가 처리될 때, 연합 서버는 템플리트가 맵핑되는 데이터 소스 함수를 호출하고, 그 데이터 유형이 템플리트 정의의 RETURNS 부분에 해당하는 값을 리턴할 것입니다. 55 페이지의 『함수 맵핑, 함수 템플리트 및 함수 맵핑 옵션』에서 자세한 정보를 참조하십시오.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

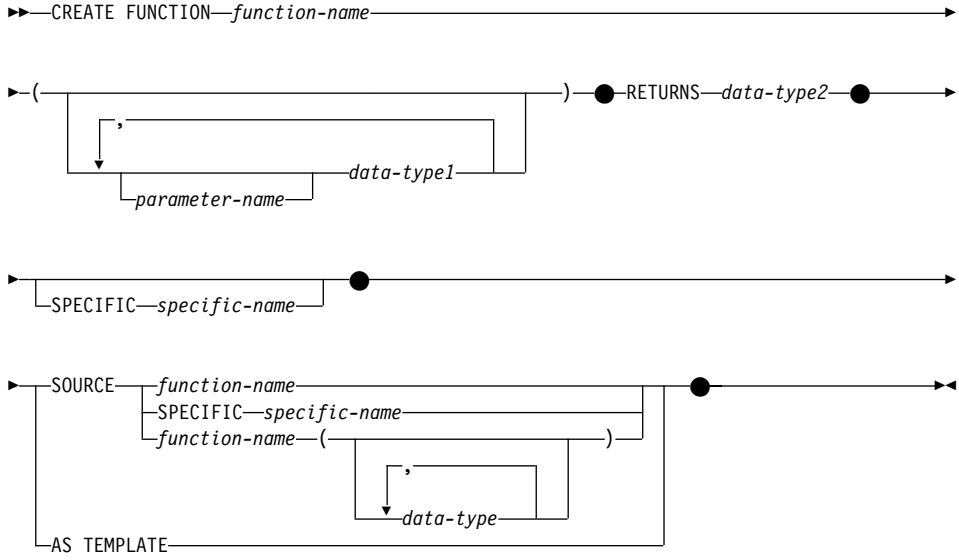
명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 함수의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 함수의 스키마 이름이 존재할 경우, 스키마에 대한 CREATEIN 특권

권한 부여 ID가 작업을 수행할 권한이 부족한 경우, 오류(SQLSTATE 42502)가 발생합니다.

SOURCE절에서 참조되는 함수에서는 권한이 필요하지 않습니다.

구문



설명

함수 이름

함수 또는 정의될 함수 템플리트 이름. 이 이름은 함수를 지시하는 규정화되거나 규정화되지 않는 이름입니다. *함수 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18인)입니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다.

매개변수 개수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고), 내재적 또는 명시적 규정자를 포함하여, 이름은 카탈로그에 설명된 함수나 함수 템플리트를 식별하지 않아야 합니다(SQLSTATE 42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정화되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 된 이름이 지정되는 경우, *스키마 이름*은 "SYS"로 시작할 수 없습니다. 그렇지 않으면, 오류(SQLSTATE 42939)가 발생합니다.

술어에서 키워드로 사용되는 이름 중 일부는 시스템에서 사용할 수 있도록 예약되어 있어서, *함수 이름*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 등이 이러한 이름이며, 217 페이지의 『기본 술어』에 기술된 비교 연산자도 여기에 포함됩니다.

사용자 정의 구별 유형을 갖는 함수를 지원하기 위해 기존 함수를 근거로 하는 사용자 정의 함수를 명명할 때, 근거가 되는 함수와 같은 이름이 사용될 수 있습니다. 이로 인해, 사용자는 추가 정의가 필요하다는 것을 인식하지 않고 사용자가 정의하는 구별 유형을 갖는 동일한 함수를 사용할 수 있습니다. 일반적으로, 함수 시그니처에 약간의 차이가 있으면, 둘 이상의 함수에 동일한 이름을 사용할 수 있습니다.

(데이터 유형,...)

함수 또는 함수 템플리트의 입력 매개변수 개수를 식별하고, 각 매개변수의 데이터 유형을 지정합니다. 함수 또는 함수 템플리트 받기를 기대하는 각 매개변수에 대해 목록에서 한 항목만 지정할 수 있습니다. 매개변수 개수는 90개를 초과할 수 없습니다. 이 한계를 초과하면, 오류(SQLSTATE 54023)가 발생합니다.

매개변수가 없는 함수 또는 함수 템플리트를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예:

```
CREATE FUNCTION WOOFER() ...
```

스키마 내에서 동일하게 명명된 두 함수 또는 함수 템플리트는 모든 해당되는 매개변수에 대해 완전히 같은 유형을 가질 수 없습니다(이 제한사항은 같은 이름을 가지고 있는 스키마 내의 함수 및 함수 템플리트에도 적용됩니다.). 길이, 정밀도 및 스케일은 이 유형의 비교에서 고려되지 않습니다. 그러므로, CHAR(8) 및 CHAR(35)는 같은 유형으로 간주되고, DECIMAL(11,2) 및 DECIMAL (4,3)도 마찬가지입니다. DECIMAL 및 NUMERIC과 같이 이러한 목적을 위해 동일한 유형으로 처리되도록 하는 유형들이 더 있습니다. 시그니처가 중복되면 SQL 오류(SQLSTATE 42723)가 발생합니다.

예를 들어, 다음 명령문이 제공되면,

```
CREATE FUNCTION PART (INT, CHAR(15)) ...
CREATE FUNCTION PART (INTEGER, CHAR(40)) ...


CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

중복되는 함수로 간주되어 두 번째 및 네번째 명령문이 실패합니다.

매개변수 이름

이 함수에서 다른 모든 매개변수의 이름과 구별되는, 매개변수에 대한 선택적 이름을 지정합니다.

데이터 유형1

매개변수의 데이터 유형을 지정합니다.

SOURCE절에 표시된 함수의 해당 매개변수 유형으로 변환할 수 있다면 유효한 모든 SQL 데이터 유형을 전래 스칼라 함수에 사용할 수 있습니다(변환 가능에 대한 106 페이지의 『데이터 유형간의 변환』에서 정의를 참조하십시오.). REF(*유형 이름*) 데이터 유형은 매개변수의 데이터 유형으로 지정할 수 있습니다(SQLSTATE 42997).

함수가 전래되었으므로, 매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다(지정할 수도 있음). 대신, 빈 괄호가 사용됩니다(예를 들어 CHAR()을 사용할 수 있음). *매개변수화된 데이터 유형*은 특정 길이, 정밀도 또는 스케일을 사용하여 정의할 수 있는 데이터 유형 중 하나입니다. 매개변수화된 데이터 유형은 문자열 데이터 유형과 십진수 데이터 유형입니다.

RETURNS

이 필수 절은 함수 또는 함수 템플리트의 출력을 식별합니다.

데이터 유형2

출력의 데이터 유형을 지정합니다.

소스 함수의 결과 유형에서 변환할 수 있다면 구별 유형에서와 마찬가지로 유효한 SQL 데이터 유형이면 유효합니다(변환 가능의 정의에 대해서는 106 페이지의 『데이터 유형간의 변환』에서 참조하십시오.).

위에 설명된 근거가 되는 함수의 매개변수에서처럼, 매개변수화된 유형의 매개변수를 지정할 필요가 없습니다. 그 대신, 빈 괄호를 사용해야 합니다 (예: VARCHAR()).

함수가 다른 함수의 기준이 될 때 RETURNS절의 데이터 유형 스펙에 적용되는 추가 고려사항 및 규칙에 대해서는 721페이지를 참조하십시오.

SPECIFIC *특정 이름*

정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. *특정 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18인)입니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다. 암시적 또는 명시적 규정자를 포함하는 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스를 식별해야 합니다. 그렇지 않으면 오류(SQLSTATE 42710)가 발생합니다.

*고유 이름*은 기존의 *함수 이름*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *함수 이름*에 대해 사용되었던 규정자가 사용됩니다. 규정자가 지정되면, *함수 이름*의 명시적 또는 암시적 규정자와 같아야 합니다. 그렇지 않으면 오류(SQLSTATE 42882)가 발생합니다.

*특정 이름*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

SOURCE

작성되는 함수가 데이터베이스 관리 프로그램에 대해 이미 알려진 다른 함수(원시 함수)에 의해 구현됨을 지정합니다. 소스 함수는 내장 함수[71]나 이전에 작성된 사용자 정의 스칼라 함수입니다.

SOURCE절은 스칼라 함수나 컬럼 함수에만 지정할 수 있습니다. 테이블 함수에는 지정할 수 없습니다.

---

71. COALESCE, NODENUMBER, NULLIF, PARTITION, TYPE_ID, TYPE_NAME, TYPE_SCHEMA 및 VALUE를 예외입니다.

SOURCE절은 다른 함수에 대한 식별을 제공합니다.

함수 이름

소스로서 사용될 특정 함수를 식별합니다. 이것은 이 *함수 이름*을 갖는 스키마에 정확히 하나의 특정 함수가 있을 경우에만 유효합니다. 이 구문 변이는 내장 함수인 원시 함수에 대해서는 유효하지 않습니다.

규정화되지 않은 이름이 제공되면, 권한 부여 ID의 현재 SQL 경로(CURRENT PATH 특수 레지스터의 값)가 함수를 찾는데 사용됩니다. 이 이름의 함수를 갖는 함수 경로의 첫번째 스키마가 선택됩니다.

명명된 스키마에 이 이름의 함수가 전혀 없거나, 그 이름이 규정화되지 않았고 이 이름의 어떤 함수도 함수 경로에 없을 경우, 오류(SQLSTATE 42704)가 발생합니다. 명명되거나 위치된 스키마에 함수에 대한 두 개 이상의 특정 인스턴스가 있으면, 오류(SQLSTATE 42725)가 발생합니다.

SPECIFIC *특정 이름*

함수 작성시 지정되거나 기본값인 *특정 이름*을 사용하여 소스로 사용할 특수한 사용자 정의 함수를 식별합니다. 이 구문 변이는 내장 함수인 원시 함수에 대해서는 유효하지 않습니다.

규정화되지 않은 이름을 제공할 경우, 권한 부여 ID의 현재 SQL 경로가 함수를 찾는데 사용됩니다. 이 특정 이름의 함수를 갖는 함수 경로의 첫번째 스키마가 선택됩니다.

명명된 스키마에 이 *특정 이름*을 가진 함수가 없거나, 이름이 규정되지 않았으며 SQL 경로에 이 *특정 이름*을 가진 함수가 없으면, 오류(SQLSTATE 42704)가 발생합니다.

함수 이름(데이터 유형,...)

원시 함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 이것은 내장 함수인 원시 함수에 대한 유일하게 유효한 구문 변이입니다.

함수 차수 규칙(168 페이지의 『함수 차수』에 설명된 대로)은 SOURCE절에 지정된 데이터 유형이 제공되는 같은 함수 이름이 같은 함수 중에서 하나의 함수를 선택할 때 적용됩니다. 그러나, 선택된 함수의 각 매개변수에 대한 데이터 유형은 원시 함수에 지정된 해당되는 데이터 유형과 정확히 같은 유형을 갖고 있어야 합니다.

함수 이름

원시 함수의 함수 이름을 제공합니다. 규정화되지 않은 이름을 제공할 경우, 사용자의 SQL 경로에 있는 스키마가 사용됩니다.

데이터 유형

해당 위치의 CREATE FUNCTION문에 지정한 데이터 유형과 일치해야 합니다(쉼표로 구분).

매개변수화된 데이터 유형에 대한 정밀도 또는 스케일, 길이를 지정하는 것은 필요하지 않습니다. 데이터 유형 일치사항을 찾을 때 이 속성들이 무시됨을 나타내기 위해 빈 괄호 집합 대신 코딩될 수 있습니다. 예를 들어, DECIMAL()은 데이터 유형이 DECIMAL(7,2)로 정의된 매개변수와 일치해야 합니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용될 수 없습니다(SQLSTATE 42601).

그러나, 길이, 정밀도 또는 스케일을 적어넣는 경우, 그 값은 CREATE FUNCTION문에 지정된 값과 정확히 일치해야 합니다. 이것은 정확하게 기대한 함수가 반드시 사용될 경우에 유용할 수 있습니다. 데이터 유형에 대한 동의어도 일치하는 것으로 간주됩니다(예를 들어, DEC와 NUMERIC는 일치함).

0 <n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치(Matching)는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 시그니처의 어떤 함수도 명명되거나 암시된 스키마에 존재하지 않을 경우, 오류(SQLSTATE 42883)가 발생합니다.

AS TEMPLATE

이 명령문이 실행 가능 코드를 가진 함수가 아니라, 함수 템플리트를 작성하는 데 사용될 것임을 나타냅니다.

규칙

- 편의상, 이 절에서는 허용되는 세 가지 구문 중 어떤 것이 SF를 식별하는 데 사용되었는지에 관계없이 CF를 작성하는 함수와 SOURCE절 SF에 식별된 함수를 호출합니다.
  - CF의 규정화되지 않은 이름과 SF의 규정화되지 않은 이름은 다를 수 있습니다.
  - 다른 함수의 소스로 명명된 함수는 그 자체가 소스로서 다른 함수를 사용할 수 있습니다. 간접적으로 호출된 함수가 오류를 발생하면 응용프로그램을 수정하기가 매우 어려울 수 있으므로, 이 기능을 이용할 때 주의해야 합니다.
  - 다음 절은 SOURCE절과 함께 지정하면 유효하지 않습니다(CF가 SF로부터 속성들을 이어받기 때문에).
    - CAST FROM . ..,
    - EXTERNAL ...,
    - LANGUAGE ...,
    - PARAMETER STYLE ...,
    - DETERMINISTIC / NOT DETERMINISTIC,
    - FENCED / NOT FENCED,
    - RETURNS NULL ON NULL INPUT / CALLED ON NULL INPUT
    - EXTERNAL ACTION / NO EXTERNAL ACTION
    - NO SQL
    - SCRATCHPAD / NO SCRATCHPAD
    - FINAL CALL / NO FINAL CALL
    - RETURNS TABLE (...)
    - CARDINALITY ...
    - ALLOW PARALLEL / DISALLOW PARALLEL
    - DBINFO / NO DBINFO

  이 규칙을 위반하면 오류(SQLSTATE 42613)가 발생합니다.
- CF의 입력 매개변수 개수는 SF에서의 개수와 같아야 합니다. 그렇지 않으면, 오류(SQLSTATE 42624)가 발생합니다.

- 다음의 경우에 CF에서 매개변수화된 데이터 유형에 대해 길이, 정밀도 또는 스케일을 지정하지 않아도 됩니다.
  - 함수의 입력 매개변수,
  - RETURNS 매개변수

  그 대신, 길이/정밀도/스케일이 원시 함수와 같음을 나타내기 위해 데이터 유형의 부분으로 빈 괄호를 지정하거나(예: VARCHAR()) 변환에 의해 빈 괄호가 판별될 수 있습니다.

  그러나, 길이, 정밀도 또는 스케일이 지정되면, 입력 매개변수와 리턴 값에 대해 아래에 간략히 설명된 대로 SF의 해당 값에 대해 CF의 값이 검사됩니다.
- CF 입력 매개변수의 스펙은 SF 입력 매개변수의 스펙에 대해 점검됩니다. CF의 각 매개변수의 데이터 유형은 SF의 해당 매개변수에 대한 데이터 유형과 같거나 *변환가능*해야 합니다. 변환 가능에 대한 정의에 대해서는 106 페이지의 『데이터 유형간의 변환』에서 참조하십시오. 매개변수가 같은 유형이 아니거나 변환 가능하지 않으면, 오류(SQLSTATE 42879)가 발생합니다.

  이 규칙은 CF 사용시 발생하는 오류에 대해서는 보장되지 않는다는 점에 유의하십시오. 데이터 유형 및 CF 매개변수의 길이 또는 정밀도와 일치하는 인수는 해당 SF 매개변수의 길이가 더 짧고, 그 정밀도가 더 낮을 경우에는 지정할 수 없는 경우도 있습니다. 일반적으로 CF 매개변수에는 해당 SF 매개변수의 속성보다 더 큰 길이 또는 정밀도 속성이 있으면 안됩니다.
- CF의 RETURNS 데이터 유형에 대한 스펙은 SF의 이 스펙에 대해 점검됩니다. SF의 마지막 RETURNS 데이터 유형은 변환 이후에 CF의 RETURNS 데이터 유형과 같거나 변환 가능해야 합니다. 그렇지 않으면, 오류(SQLSTATE 42866)가 발생합니다.

  이 규칙은 CF 사용시 발생하는 오류에 대해서는 보장되지 않는다는 점에 유의하십시오. 데이터 유형 및 SF RETURNS 데이터 유형의 길이 또는 정밀도 속성과 일치하는 결과값은 CF RETURNS 데이터 유형의 길이가 더 짧거나 정밀도가 더 낮은 경우에 지정할 수 없는 경우도 있습니다. SF RETURNS 데이터 유형의 속성보다 적은 길이 또는 정밀도 속성을 가지므로, CF의 RETURNS 데이터 유형 지정 선택시 주의해야 합니다.

주

- 하나의 데이터 유형이 다른 데이터 유형으로 변환 가능한 지를 판별할 때 CHAR 및 DECIMAL과 같은 매개변수화된 데이터 유형에 대해 길이나 정밀도, 그리고 스케일은 고려되지 않습니다. 그러므로, 함수를 사용할 때 원시 데이터 유형의 값을 목표 데이터 유형의 값으로 변환하려고 하면 오류가 발생할 수 있습니다. 예를 들어, VARCHAR은 DATE로 변환 가능하나 원시 유형이 실제로 VARCHAR(5)로 정의되면, 함수를 사용할 때 오류가 발생합니다.
- 사용자 정의 함수의 매개변수에 대한 데이터 유형을 선택할 때, 그 입력 값에 영향을 주게 될 승격 규칙을 고려하십시오(104 페이지의 『데이터 유형의 승격』 참조). 예를 들어, 입력 값으로 사용된 상수가 예상한 것과 다른 내장 데이터 유형을 가질 수도 있고, 좀더 심각하게는 예상한 데이터 유형으로 승격되지 않을 수 있습니다. 승격 규칙에 따라, 매개변수에 다음의 데이터 유형을 사용하는 것이 좋습니다.
  - SMALLINT 대신 INTEGER
  - REAL 대신 DOUBLE
  - CHAR 대신 VARCHAR
  - GRAPHIC 대신 VARGRAPHIC
- 아직 존재하지 않는 스키마 이름을 사용하여 함수를 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 연합 서버가 데이터 소스 함수를 인식하기 위해서는, 함수가 연합 데이터베이스에 있는 상대편에 맵핑되어야 합니다. 데이터베이스에 상대편이 없으면, 사용자가 상대편을 작성한 후 맵핑을 작성해야 합니다.

  상대편은 함수(스칼라나 소스) 또는 함수 템플리트일 수 있습니다. 사용자가 함수나 필요한 맵핑을 작성하면, 함수를 지정하는 조회가 처리될 때마다 DB2가 (1) 호출하는 전략을 데이터 소스 함수를 호출하는 전략과 비교하고, (2) 오버헤드가 덜 들어가는 것으로 예상되는 함수를 호출합니다.

  사용자가 함수 템플리트와 맵핑을 작성하고 나면, 그 템플리트를 지정하는 조회가 처리될 때마다 DB2는 맵핑되는 데이터 소스 함수를 호출합니다(이 함수를

호출하기 위한 액세스 플랜이 존재할 경우). 연합 시스템에서 함수 호출에 드는 오버헤드를 제어하는 데 대한 *응용프로그램 개발 안내서*에서 좀더 자세한 정보를 참조하십시오.

예

예 1: Pellow의 원래 CENTRE 외부 스칼라 함수를 작성한 이후에, 다른 사용자가 그 함수를 작성하려고 합니다(단, 이 함수는 정수 인수만을 승인하도록 고안되었음).

```
CREATE FUNCTION MYCENTRE (INTEGER, INTEGER)
    RETURNS FLOAT
    SOURCE  PELLOW.CENTRE (INTEGER, FLOAT)
```

예 2: 내장 INTEGER 데이터 유형에 기초하여 구별 유형 HATSIZE을 작성하였고, 서로 다른 부서의 평균 모자 크기를 계산하려면 AVG 함수가 있는 것이 유용하다는 것을 알게 되었습니다. 다음과 같이 간편하게 수행할 수 있습니다.

```
CREATE FUNCTION AVG (HATSIZE) RETURNS (HATSIZE)
    SOURCE SYSIBM.AVG (INTEGER)
```

구별 유형을 작성하면 필수 유형변환 함수가 작성되며 인수에 대해 HATSIZE에서 INTEGER로 그리고 INTEGER에서 HATSIZE로의 변환이 허용됩니다.

예 3: 연합 시스템에서, 사용자가 이중 정밀도 부동 소수점의 형태로 값을 형성하는 테이블 통계를 리턴하는 Oracle UDF를 호출하고 싶어합니다. 연합 서버는 함수와 연합 데이터베이스 상대편간의 맵핑이 있는 경우에만 이 함수를 인식할 수 있습니다. 그러나 그러한 상대편이 존재하지 않습니다. 사용자는 함수 템플리트의 형태로 하나 제공하고 NOVA라고 하는 스키마에 이 템플리트를 지정하기로 결정합니다. 사용자는 다음 코드를 사용하여 템플리트를 연합 서버에 등록합니다. 맵핑을 위한 사용자 코드에 대해서는 739 페이지의 『예』에서 참조하십시오.

```
CREATE FUNCTION NOVA.STATS (DOUBLE, DOUBLE)
     RETURNS DOUBLE
     AS TEMPLATE
```

예 4: 연합 시스템에서, 사용자가 특정 조직의 직원이 보너스로서 버는 달러 금액을 리턴하는 Oracle UDF를 호출하고 싶어합니다. 연합 서버는 함수와 연합 데이터베이스 상대편간의 맵핑이 있는 경우에만 이 함수를 인식할 수 있습니다. 그러

한 상대편이 존재하지 않습니다. 따라서 사용자는 함수 템플리트의 형태로 하나 작성합니다. 사용자는 다음 코드를 사용하여 이 템플리트를 연합 서버에 등록합니다. 맵핑을 위한 사용자 코드에 대해서는 739 페이지의 『예』에서 참조하십시오.

```
CREATE FUNCTION BONUS ()
     RETURNS DECIMAL (8,2)
     AS TEMPLATE
```

CREATE FUNCTION (SQL 스칼라, 테이블 또는 행)

이 명령문은 사용자 정의 SQL 스칼라, 테이블 또는 행 함수를 정의하는데 사용됩니다. *스칼라 함수*는 호출될 때마다 단일 값을 리턴하며, 일반적으로 SQL 표현식이 유효하면 이 함수도 유효합니다. *테이블 함수*는 FROM절에서 사용될 수도 있으며 테이블을 리턴합니다. *행 함수*는 전송 함수로 사용될 수도 있으며 한 행을 리턴합니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

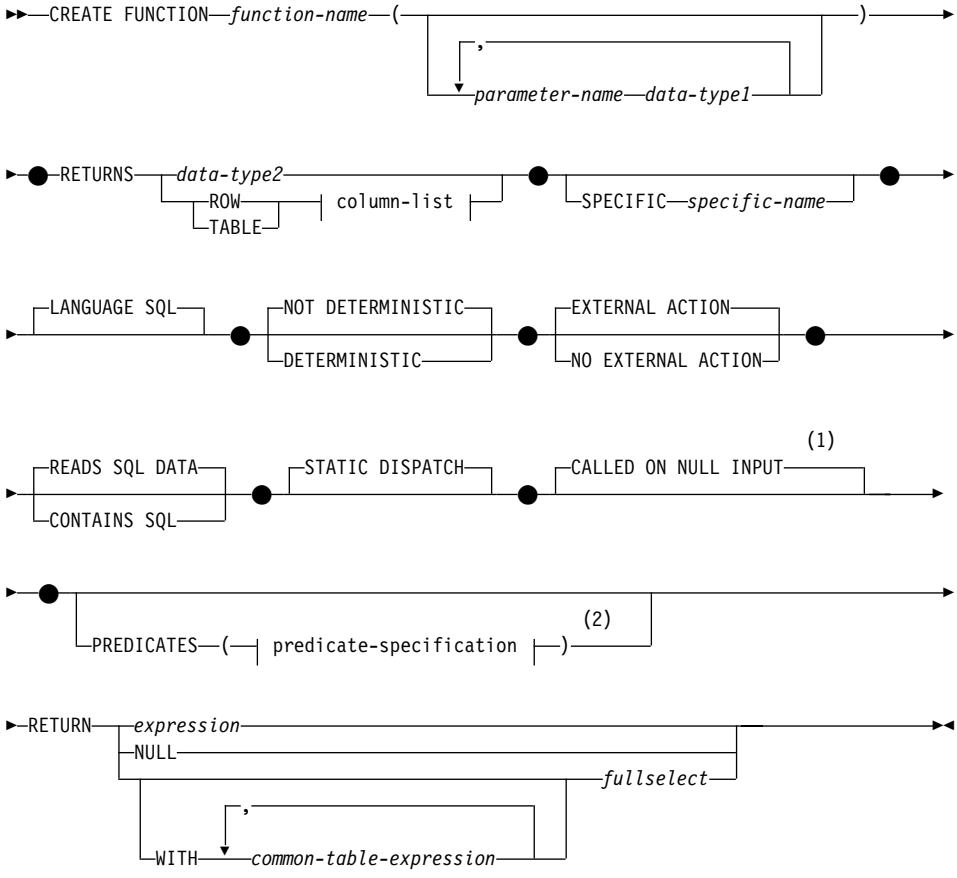
- SYSADM 또는 DBADM 권한
- 함수의 스키마 이름이 기존의 스키마를 가리키지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 함수의 스키마 이름이 기존의 스키마를 가리키지 않을 경우, 데이터베이스에 대한 CREATEIN 특권

명령문의 권한 부여 ID가 SYSADM 또는 DBADM 권한을 가지고 있지 않고, 함수가 테이블이나 뷰를 식별할 경우, 명령문의 권한 부여 ID가 보유하는 특권에는 (고려하는 GROUP 특권 없이) 식별된 각 테이블과 뷰에 대한 SELECT WITH GRANT OPTION이 포함되어야 합니다.

함수 정의자가 SYSADM 권한을 가지고 있어서 함수를 작성만 할 수 있을 경우, 그 정의자에게는 함수 작성을 위한 내재된 DBADM 권한이 부여됩니다.

권한 부여 ID가 작업을 수행할 권한이 부족한 경우, 오류(SQLSTATE 42502)가 발생합니다.

구문



column-list:

주:

1 CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있습니다.

2 RETURNS가 스칼라 결과(데이터 유형2)를 지정할 경우에만 유효합니다.

설명

함수 이름

정의되는 함수를 명명합니다. 이 이름은 함수를 지시하는 규정화되거나 규정화되지 않는 이름입니다. *함수 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18인)입니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다.

매개변수 개수와 각 매개변수의 데이터 유형과 함께(데이터 유형의 길이, 정밀도 또는 스케일 속성은 고려하지 않고), 내재적 또는 명시적 규정자를 포함하여, 이름은 카탈로그에 설명된 함수를 식별하지 않아야 합니다(SQLSTATE 42723). 매개변수의 개수 및 유형과 함께(스키마 내에서 고유한), 규정화되지 않은 이름은 스키마에서 고유하지 않아도 됩니다.

두 부분으로 이루어진 이름이 지정된 경우, *스키마 이름*은 "SYS"로 시작될 수 없습니다(SQLSTATE 42939).

술어에서 키워드로 사용되는 이름 중 일부는 시스템에서 사용할 수 있도록 예약되어 있어서, *함수 이름*으로 사용할 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 등이 이러한 이름이며, 217 페이지의 『기본 술어』에 기술된 비교 연산자도 여기에 포함됩니다.

함수 시그니처에 약간의 차이가 있으면, 둘 이상의 함수에 동일한 이름을 사용할 수 있습니다. 이것에 대한 제한이 없다고 하여도, 외부 사용자 정의 테이블 함수는 내장 함수와 이름이 같으면 안 됩니다.

매개변수 이름

이 함수에서 다른 모든 매개변수의 이름과 구별되는 이름.

데이터 유형1

매개변수의 데이터 유형을 지정합니다.

- CREATE TABLE문의 *데이터 유형1* 정의에서 지정할 수 있는 SQL 데이터 유형 스펙 및 약어.

- REF를 지정할 수는 있지만, 그 REF의 범위는 지정되지 않습니다. 시스템은 매개변수나 결과의 범위를 판단하려고 하지 않습니다. 함수 내용에서, 참조 유형은 범위를 수반하도록 처음으로 유형변환을 수행할 경우에만 참조 취소 조작에서 사용할 수 있습니다. 마찬가지로, SQL 함수에서 리턴되는 참조는 범위를 수반하도록 처음으로 유형변환을 수행할 경우에만 참조 취소 조작에서 사용할 수 있습니다.
- LONG VARCHAR 및 LONG VARGRAPHIC 데이터 유형은 사용될 수 없습니다(SQLSTATE 42815).

RETURNS

이 필수 절은 함수의 출력 유형을 식별합니다.

데이터 유형2

출력의 데이터 유형을 지정합니다.

이 명령문에서, 함수 매개변수에 대해 *데이터 유형1*에서 설명한 SQL 함수의 매개변수에 대해 적용한 것과 같은 고려사항이 적용됩니다.

ROW *컬럼 목록*

이 함수의 결과는 단일 행입니다. 함수가 여러 행을 리턴할 경우, 오류가 발생합니다(SQLSTATE 21505). *컬럼 목록*에는 최소한 두 개의 컬럼이 포함되어야 합니다(SQLSTATE 428F0).

행 함수는 구조화 유형에 대한 변환 함수로만 사용할 수 있습니다(해당되는 매개변수로 하나의 구조화 유형을 수반하고 기본 유형만 리턴합니다).

TABLE *컬럼 목록*

이 함수의 결과는 테이블입니다.

컬럼 목록

ROW 또는 TABLE 함수에 대해 리턴되는 컬럼 이름 및 데이터 유형 목록

컬럼 이름

이 컬럼의 이름을 지정합니다. 이름을 규정화할 수 없으며 행의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다.

데이터 유형3

컬럼의 데이터 유형을 지정하며, SQL 함수의 매개변수에서 지원되는 데이터 유형이면 됩니다.

SPECIFIC *특정 이름*

정의되는 함수의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 이 함수를 근거로 할 때, 함수를 삭제할 때 또는 함수에 주석을 붙일 때 사용할 수 있습니다. 이 이름은 함수를 호출할 경우에는 사용할 수 없습니다. *특정 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18)입니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다. 내재적 또는 명시적 규정자를 포함하여, 이름은 응용프로그램 서버에 존재하는 다른 함수 인스턴스를 식별하지 않아야 합니다. 그렇지 않으면 오류가 발생합니다 (SQLSTATE 42710).

*고유 이름*은 기존의 *함수 이름*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *함수 이름*에 대해 사용되었던 규정자가 사용됩니다. 규정자가 지정되면, *함수 이름*의 명시적 또는 내재적 규정자와 같아야 합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42882).

*특정 이름*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmssxxx입니다.

LANGUAGE SQL

함수가 SQL을 사용하여 작성됨을 지정합니다. 지원되는 SQL은 현재 RETURN 문으로 제한됩니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**

이 선택적 절은 함수가 항상 주어진 값에 대해 동일한 결과를 리턴하는지 (DETERMINISTIC) 또는 함수가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다 즉, DETERMINISTIC 함수는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC를 지정하여 방지할 수 있습니다.

함수의 내용이 특수 레지스터에 액세스하거나 다른 비결정 함수를 호출할 경우 NOT DETERMINISTIC을 명시적이나 내재적으로 지정해야 합니다 (SQLSTATE 428C2).

NO EXTERNAL ACTION 또는 EXTERNAL ACTION

이 선택적 절은 데이터베이스 관리 프로그램에 의해 관리되지 않는 오브젝트의 상태를 변경하는 일부 조치를 함수에서 사용하고 있는 지를 지정합니다. NO EXTERNAL ACTION을 지정하면, 시스템은 함수에 외부적 영향이 없는 것으로 간주되는 특정 최적화를 사용할 수 있습니다.

함수의 내용이 외부 조치가 있는 또 다른 함수를 호출할 경우 EXTERNAL ACTION을 명시적이나 내재적으로 지정해야 합니다(SQLSTATE 428C2).

READS SQL DATA 또는 CONTAINS SQL

실행될 수 있는 SQL문 유형을 나타냅니다. 지원되는 SQL문은 RETURN문이므로, 표현식이 부속 조회인지의 여부에 따라 구별해야 합니다.

READS SQL DATA

SQL 데이터를 수정하지 않는 SQL문이 함수에 의해 실행될 수 있음을 나타냅니다(SQLSTATE 42985). 별명이나 OLEDB 테이블 함수는 SQL문에서 참조될 수 없습니다(SQLSTATE 42997).

CONTAINS SQL

SQL 데이터를 읽지도, 수정하지도 않는 SQL문이 함수에 의해 실행될 수 있음을 나타냅니다(SQLSTATE 42985).

STATIC DISPATCH

이 선택적 절은 함수 결정 시, DB2가 함수 매개변수의 정적 유형(선언된 유형)을 기초로 함수를 선택함을 나타냅니다.

CALLED ON NULL INPUT

이 절은 해당되는 인수 중 어느 하나가 널(NULL)인지에 관계 없이 함수가 호출됨을 나타냅니다. 이 절은 널(NULL) 값이나 널(NULL) 이외의 값을 리턴할 수 있습니다. 널(NULL) 인수 값에 대한 테스트 책임은 사용자 정의 함수에 있습니다.

CALLED ON NULL INPUT 대신 NULL CALL을 사용할 수 있습니다.

PREDICATES

이 함수를 사용하는 술어의 경우, 이 절은 색인 확장을 보여줄 수 있는 술어들을 식별하고, 그 술어의 검색 조건에 대해 선택적 SELECTIVITY 절을 사용할 수 있습니다. PREDICATES 절을 지정할 경우, 함수는 NO EXTERNAL ACTION의 DETERMINISTIC으로 정의되어야 합니다(SQLSTATE 42613).

술어 스펙

술어 스펙에 대해서는 655 페이지의 『CREATE FUNCTION(외부 스칼라)』에서 자세한 내용을 참조하십시오.

RETURN

함수의 리턴값을 지정합니다. 매개변수 이름은 RETURN문에서 참조될 수 있습니다. 매개변수 이름은 애매한 참조를 피하기 위해 함수 이름으로 규정될 수 있습니다.

표현식

함수에 대해 리턴될 표현식을 지정합니다. 표현식의 결과 데이터 유형은 RETURNS 절에 정의된 데이터 유형에 지정할 수 있어야 합니다(저장 지정 규칙을 사용하여)(SQLSTATE 8166). 스칼라 표현식(스칼라 fullselect 이외의)은 테이블 함수에 대해 지정할 수 없습니다(SQLSTATE 428F1).

NULL

함수가 RETURNS 절에 정의된 데이터 유형의 널(NULL) 값을 리턴함을 지정합니다.

WITH *공통 테이블 표현식*

뒤에 나올 fullselect과 함께 사용할 공통 테이블 표현식을 정의합니다. 483 페이지의 『공통 테이블 표현식』에서 참조하십시오.

fullselect

함수에 대해 리턴될 행을 지정합니다. fullselect의 컬럼 수는 함수 결과에서의 컬럼 수와 일치해야 하며(SQLSTATE 42811), fullselect의 정적 컬럼 유형은 컬럼에 대한 지정 규칙을 사용하여, 함수 결과의 선언된 컬럼 유형에 대해 지정 가능해야 합니다(SQLSTATE 42866).

함수가 스칼라 함수일 경우, fullselect는 하나의 컬럼(SQLSTATE 42823)과, 최대 한 행(SQLSTATE 21000)을 리턴해야 합니다.

함수가 행 함수이면, 최대 한 행을 리턴해야 합니다(SQLSTATE 21505).

함수가 테이블 함수일 경우, 하나 이상의 컬럼과 함께 0개 이상의 행을 리턴할 수 있습니다.

주

- 함수 내용 내에서의 함수 호출 분석은 CREATE FUNCTION문에 대해 효율적인 함수 경로에 따라 수행되고 그 함수가 작성되고 나면 변경되지 않습니다.
- SQL 함수에 날짜 및 시간 특수 레지스터 중 하나에 대한 여러 참조가 포함될 경우, 모든 참조는 같은 값을 리턴하며, 이것은 그 함수를 호출한 명령문에서의 레지스터 호출에 의해 리턴된 것과 같은 값이어야 합니다.
- SQL 함수의 내용에는 자체나, 이를 호출하는 다른 함수 또는 메소드에 대한 순환 호출이 포함될 수 없습니다.
- 다음 규칙은 함수나 메소드를 작성하는 모든 명령문에 의해 시행됩니다.
  - 함수는 메소드와 같은 시그니처를 가질 수 없습니다(함수의 첫번째 *매개변수 유형*을 메소드의 *주제 유형*과 비교하여).
  - 함수 및 메소드는 겹쳐쓰는 관계에 있을 수 없습니다. 즉, 함수가, 첫번째 매개변수가 주제인 메소드일 경우, 다른 메소드를 겹쳐쓰기하거나 다른 메소드에 의해 겹쳐쓰기되어서는 안 됩니다.
  - 겹쳐쓰기는 함수에 적용되지 않으므로, 두 함수가 메소드여서 하나가 다른 메소드를 겹쳐쓰기할 경우와 같이 두 함수가 존재할 수 있습니다.

  위의 규칙에서의 매개변수 유형 비교를 위해,
  - 매개변수 이름, 길이, AS LOCATOR 및 FOR BIT DATA가 무시됩니다.
  - 부속 유형은 해당되는 수퍼 유형과 다른 것으로 간주됩니다.

예

예 1: 기존의 사인 및 코사인 함수를 사용하여 값의 탄젠트를 리턴하는 스칼라 함수를 정의합니다.

```
CREATE FUNCTION TAN (X DOUBLE)
   RETURNS DOUBLE
   LANGUAGE SQL
```

```
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN SIN(X)/COS(X)
```

예 2: 구조화 유형 PERSON에 대한 변환 함수를 정의합니다.

```
CREATE FUNCTION FROMPERSON (P PERSON)
  RETURNS ROW (NAME VARCHAR(10), FIRSTNAME VARCHAR(10))
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN VALUES (P..NAME, P..FIRSTNAME)
```

예 3: 지정된 부서 번호의 사원들을 리턴하는 테이블 함수를 정의합니다.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN
     SELECT EMPNO, LASTNAME, FIRSTNME
       FROM EMPLOYEE
       WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
```

이 함수의 정의자는 EMPLOYEE 테이블에 대해 SELECT WITH GRANT OPTION 특권을 가지고 있어야 하고 모든 사용자는 각 부서 번호에 대한 결과 컬럼의 데이터에 효율적으로 액세스할 수 있도록 테이블 함수 DEPTEMPLOYEES를 호출할 수 있습니다.

CREATE FUNCTION MAPPING

CREATE FUNCTION MAPPING문은 다음을 수행하는 데 사용됩니다.

- 연합 데이터베이스 함수나 함수 템플리트와 데이터 소스 함수 사이에 맵핑을 작성하십시오. 맵핑은 연합 데이터베이스 함수나 템플리트를 (1) 지정된 데이터 소스나 (2) 데이터 소스의 범위(예를 들어, 특정 유형 및 버전의 모든 데이터 소스)에 있는 함수와 연관시킬 수 있습니다.
- 연합 데이터베이스 함수와 데이터 소스 함수 사이의 기본 맵핑을 사용안함으로 합니다.

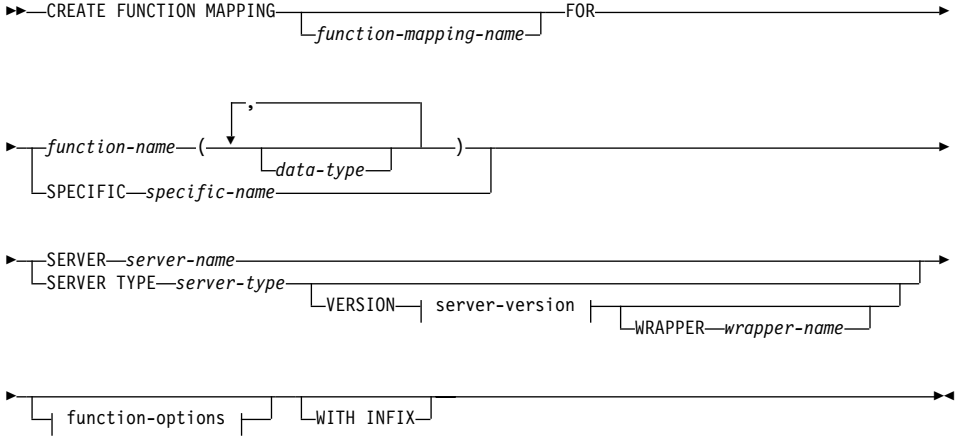
호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

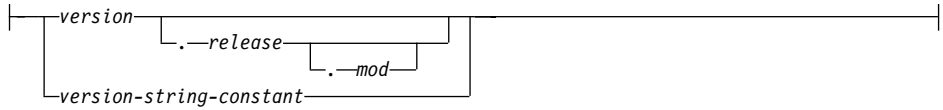
권한 부여

명령문의 권한 부여 ID는 SYSADM 또는 DBADM 권한이 있어야 합니다.

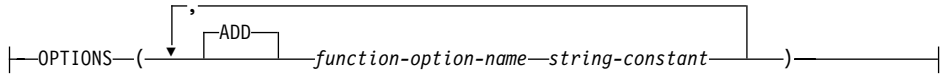
구문



server-version:



function-options:



설명

함수 맵핑 이름

함수 맵핑에 이름을 부여합니다. 이름은 카탈로그에 이미 기술되어 있는 함수 맵핑을 식별해선 안됩니다(SQLSTATE 42710).

*함수 맵핑 이름*이 생략되면, 시스템 생성 고유 이름이 지정됩니다.

함수 이름

맵핑할 함수나 함수 템플리트의 규정화된 이름 또는 규정화되지 않은 이름입니다.

데이터 유형

입력 매개변수를 가지고 있는 함수나 함수 템플리트의 경우, *데이터 유형*은 그러한 매개변수의 데이터 유형을 지정합니다. *데이터 유형*은 LONG VARCHAR, LONG VARGRAPHIC, DATALINK, 대형 오브젝트(LOB) 유형 또는 사용자 정의 유형일 수 없습니다.

SPECIFIC *특정 이름*

맵핑할 함수나 함수 템플리트를 식별합니다. 함수나 함수 템플리트가 연합 데이터베이스에서 고유한 *함수 이름*을 가지지 않는다면 *특정 이름*을 지정하십시오.

SERVER *서버 이름*

맵핑되고 있는 함수를 포함하는 데이터 소스에 이름을 부여하십시오.

TYPE *서버 유형*

맵핑되고 있는 함수를 포함하는 데이터 소스의 유형에 이름을 부여하십시오.

VERSION

*서버 유형*으로 표시된 데이터 소스의 버전을 지정합니다.

버전

버전 번호를 지정합니다. *버전*은 정수여야 합니다.

릴리스

*버전*으로 표시된 버전의 릴리스 번호를 지정합니다. *릴리스*는 정수여야 합니다.

mod

*릴리스*로 표시된 릴리스의 수정 번호를 지정합니다. *mod*는 정수여야 합니다.

버전 문자열 상수

완전한 버전 지정을 합니다. *버전 문자열 상수*는 단일 값(예를 들어, '8i') 이거나 또는 *버전*, *릴리스* 및 *mod*의 결합된 값(예를 들어, '8.0.3')일 수 있습니다.

WRAPPER *랩퍼 이름*

server-type 및 *server-version*에 의해 표시된 유형 및 버전의 데이터 소스와 상호 작용하기 위해 연합 서버가 사용하는 랩퍼의 이름을 지정합니다.

OPTIONS

어느 함수 맵핑 옵션이 작동가능회될 지를 나타냅니다. *함수 이름 옵션* 및 그 설정값에 대한 설명은 1406 페이지의 『함수 맵핑 옵션』에서 참조하십시오.

ADD

하나 이상의 함수 맵핑 옵션을 작동가능화 시킵니다.

함수 옵션 이름

함수 맵핑이나 맵핑에 포함된 데이터 소스 함수에 적용하는 함수 맵핑 옵션에 이름을 부여합니다.

문자열 상수

*함수 옵션 이름*에 대한 설정값을 문자열 상수로서 지정합니다.

WITH INFIX

데이터 소스 함수가 삽입사(infix) 포맷으로 생성되게 지정합니다.

주

- 다음과 같은 경우에 연합 데이터베이스 함수나 함수 템플리트가 데이터 소스 함수에 맵핑될 수 있습니다.
  - 연합 데이터베이스 함수나 템플리트에는 데이터 소스 함수와 동일한 갯수의 입력 매개변수가 있습니다.
  - 연합 함수나 템플리트에 정의되는 데이터 유형은 데이터 소스 함수에 정의되는 해당 데이터 유형과 호환됩니다.
- 분산 요청이 데이터 소스 함수에 맵핑하는 DB2 함수를 참조하면, 최적화 알고리즘이 요청 처리시 어느 함수든 호출하도록 전략을 개발합니다. 그렇게 하여 DB2 함수를 호출하면 데이터 소스 함수를 호출하는 것보다 오버헤드가 덜 생깁니다. 그렇지 않고, DB2 함수 호출에 더 많은 오버헤드가 필요하면 데이터 소스 함수가 호출됩니다.
- 분산 요청이 데이터 소스 함수에 맵핑하는 DB2 함수 템플리트를 참조하면, 요청 처리시 데이터 소스 함수만 호출될 수 있습니다. 템플리트는 실행 가능 코드가 없기 때문에 호출될 수 없습니다.
- 이들을 사용안함으로 하여(삭제될 수는 없음) 기본적인 함수 맵핑을 작동 불능으로 할 수 있습니다. 기본적인 함수 맵핑을 사용안함으로 하기 위해서는, 맵핑 내에서 DB2 함수의 이름을 지정하고 DISABLE 옵션을 'Y'로 설정하도록 CREATE FUNCTION MAPPING문을 코딩하십시오.
- SYSIBM 스키마에 있는 함수에는 특정 이름이 없습니다. SYSIBM 스키마에 있는 함수에 대한 기본적인 함수 맵핑을 겹쳐쓰려면, 규정자 SYSIBM 및 함수 이름(LENGTH와 같은)으로 *함수 이름*을 지정하십시오.
- 다음 조건에서는 주어진 작업 단위(UOW) 내에서 CREATE FUNCTION MAPPING문을 처리할 수 없습니다.
  - 명령문이 단일 데이터 소스를 참조하며, UOW가 이미 이 데이터 소스 내의 테이블이나 뷰에 대한 별명(nickname)을 참조하는 SELECT문을 포함합니다.
  - 명령문이 데이터 소스의 카테고리를 참조하며(예를 들어, 특정 유형 및 버전의 모든 데이터 소스), UOW가 이미 이들 데이터 소스 중 하나 안에 있는 테이블이나 뷰에 대한 별명(nickname)을 참조하는 SELECT문을 포함합니다.

예

예 1: 함수 템플리트를 모든 Oracle 데이터 소스가 액세스할 수 있는 UDF로 맵핑하십시오. 템플리트는 STATS라고 부르며 NOVA라는 스키마에 속합니다. Oracle UDF는 STATISTICS라 하며 STAR라고 하는 스키마에 속합니다.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN1
   FOR NOVA.STATS ( DOUBLE, DOUBLE )
   SERVER TYPE ORACLE
   OPTIONS ( REMOTE_NAME 'STAR.STATISTICS' )
```

예 2: BONUS라는 함수 템플리트를 ORACLE1이라는 Oracle 데이터 소스에서 사용되는 BONUS라 불리기도 하는 UDF에 맵핑하십시오.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN2
   FOR BONUS()
   SERVER ORACLE1
   OPTIONS ( REMOTE_NAME 'BONUS')
```

예 3: 연합 데이터베이스에 정의되는 WEEK 시스템 함수와 Oracle 데이터 소스에 정의되는 유사한 함수간에 기본적인 함수 맵핑이 있다고 가정하십시오. Oracle 데이터를 요청하고 WEEK를 참조하는 조회가 처리될 때, 최적화 알고리즘이 어느 것이 오버헤드가 덜 든다고 추정하는지에 따라 WEEK 또는 이것의 Oracle 상대편이 호출될 것입니다. DBA는 그러한 조회에 대해 WEEK만 호출된 경우 성능이 어떻게 영향받을지에 대해 알고 싶어합니다. 매번 WEEK이 호출되게 하려면, DBA가 맵핑을 사용안함으로 해야 합니다.

```
CREATE FUNCTION MAPPING
   FOR SYSFUN.WEEK(INT)
   TYPE ORACLE
   OPTIONS ( DISABLE 'Y' )
```

예 4: 지역 함수 UCASE(CHAR)를 ORACLE2라는 Oracle 데이터 소스에서 사용되는 UDF에 맵핑하십시오. Oracle UDF의 호출 당 추정되는 명령어 수를 포함시키십시오.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN4
   FOR SYSFUN.UCASE(CHAR)
   SERVER ORACLE2
   OPTIONS
      ( REMOTE_NAME 'UPPERCASE',
      INSTS_PER_INVOC '1000' )
```

CREATE INDEX

CREATE INDEX문은 다음을 작성하는 데 사용됩니다.

- DB2 테이블상의 색인
- 색인 스펙: 최적화 알고리즘에게 데이터 소스에 색인이 있음을 나타내는 메타 데이터

호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

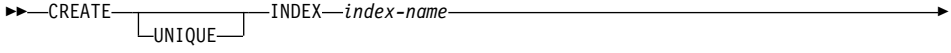
권한 부여

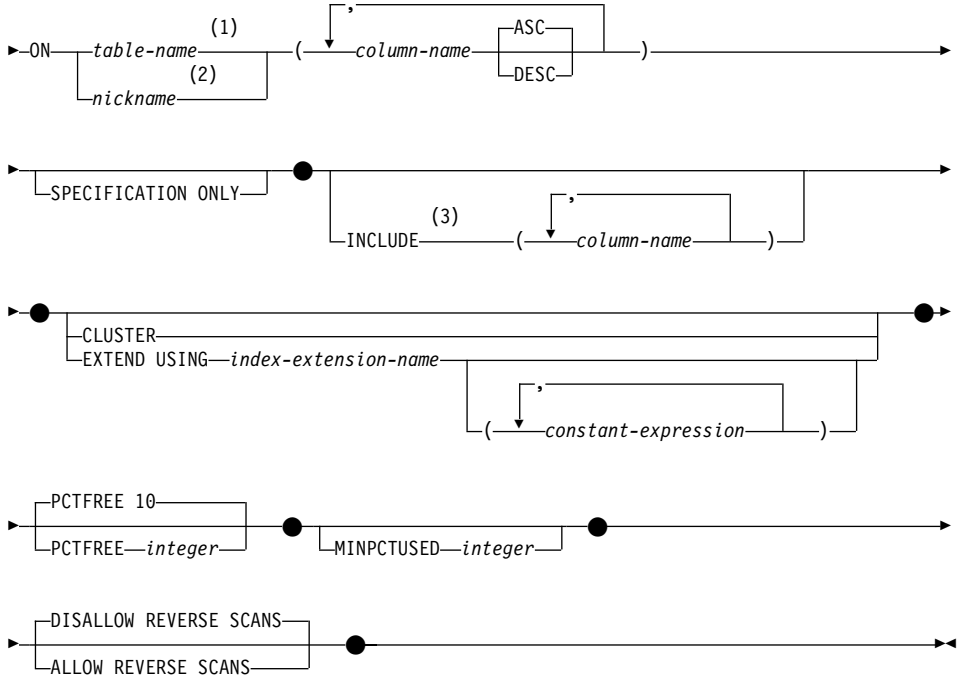
명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 다음 중 하나:
  - 테이블에 대해 CONTROL 특권
  - 테이블에서의 INDEX 특권

  다음 중 하나:
  - 색인의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 색인의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

구문





주:

1 연합 시스템에서, *테이블 이름*은 연합 데이터베이스에서 테이블을 식별해야 합니다. 데이터 소스 테이블을 식별할 수는 없습니다.

2 *별명(nickname)*이 지정되면, CREATE INDEX 명령문이 색인 스펙을 작성할 것입니다. INCLUDE, CLUSTER, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS 및 ALLOW REVERSE SCANS는 지정될 수 없습니다.

3 INCLUDE절은 UNIQUE가 지정된 경우에만 지정할 수 있습니다.

설명

UNIQUE

ON *테이블 이름*을 지정할 경우, UNIQUE는 테이블에 색인 키와 같은 값을 갖는 둘 이상의 행을 포함하지 못하도록 합니다. 고유성은 행을 갱신하거나 새로운 행을 삽입하는 SQL문의 끝에서 적용됩니다. 1451 페이지의 『부록J. 트리거와 제한조건의 상호 작용』에서 자세한 내용을 참조하십시오.

CREATE INDEX문이 실행될 때에도 고유성이 점검됩니다. 테이블에 중복되는 키 값을 갖는 행을 포함할 경우 색인은 작성되지 않습니다.

UNIQUE가 사용되면, 널(NULL) 값이 다른 값처럼 처리됩니다. 예를 들어, 키가 널(NULL) 값을 포함하는 단일 컬럼일 경우, 그 컬럼에는 두 개 이상의 널(NULL) 값을 포함할 수 없습니다.

UNIQUE 옵션이 지정되고 테이블이 파티션 키를 갖고 있을 경우, 색인 키의 컬럼은 파티션 키의 상위 세트이어야 합니다. 즉, 고유 색인 키에 대해 지정된 컬럼에는 파티션 키의 모든 컬럼이 포함되어야 합니다(SQLSTATE 42997).

ON *별명(nickname)*이 지정되면, 색인 키에 대한 데이터가 데이터 소스 테이블 의 모든 행에 대해 고유한 값을 포함하는 경우에만 UNIQUE가 지정되어야 합니다. 고유성이 점검되지 않을 것입니다.

테이블 이름은 선언된 임시 테이블이 될 수 없습니다(SQLSTATE 42995).

INDEX *색인 이름*

색인이나 색인 스펙에 이름을 붙입니다. 암시적 또는 명시적 규정자를 포함하는 이름은 카탈로그에 설명된 색인 또는 색인 스펙을 식별하지 말아야 합니다. 규정자는 SYSIBM, SYSCAT, SYSFUN 또는 SYSSTAT이면 안됩니다(SQLSTATE 42939).

ON *테이블 이름* **또는** *별명*

*테이블 이름*은 색인이 작성될 테이블에 이름을 붙입니다. 테이블은 카탈로그에 설명된 기본 테이블(뷰가 아님) 또는 요약 테이블이어야 합니다. 이것은 카탈로그 테이블(SQLSTATE 42832)이나 선언된 임시 테이블(SQLSTATE 42995)의 이름을 지정하면 안됩니다. UNIQUE가 지정되어 있고 *테이블 이름*이 입력된 테이블인 경우 서브테이블이어서는 안됩니다(SQLSTATE 429B3). UNIQUE가 지정되면, *테이블 이름*은 요약 테이블이 될 수 없습니다(SQLSTATE 42809).

*별명*은 색인 스펙이 작성될 테이블에 대한 별명입니다. *별명*은 색인 스펙에 의해 색인이 설명되는 데이터 소스나, 그러나 테이블에 기초하는 데이터 소스 뷰를 참조합니다. *별명*은 카탈로그에서 나열되어야 합니다.

컬럼 이름

색인의 경우, *컬럼 이름*은 색인 키의 일부가 될 컬럼을 식별합니다. 색인 스펙의 경우, *컬럼 이름*은 연합 서버가 데이터 소스 테이블의 컬럼을 참조하는 이름입니다.

각 *컬럼 이름*은 테이블의 컬럼을 식별하는 규정화된 이름이어야 합니다. 16개 이하의 컬럼을 지정할 수 있습니다. *테이블 이름*이 입력된 테이블인 경우, 15 컬럼 이하를 지정할 수 있습니다. *테이블 이름*이 서브테이블일 경우, 최소한 하나의 *컬럼 이름*이 서브테이블에 도입되어야 합니다, 즉, 서브테이블에서 계승되지 않아야 합니다(SQLSTATE 428DS). *컬럼 이름*은 반복될 수 없습니다(SQLSTATE 42711).

지정된 컬럼의 저장 길이의 합은 1024 이하여야 합니다. *테이블 이름*이 입력된 테이블인 경우, 색인 길이 한계는 4 바이트 단위로 추가 생성됩니다.

컬럼의 데이터 유형과 널(NULL) 입력 가능 여부에 따라 시스템 오버헤드에 의해 이 길이가 생성될 수 있습니다. 이 한도에 영향을 주는 오버헤드에 대한 853 페이지의 『바이트 계수』에서 좀더 자세한 정보를 참조하십시오.

개별적 컬럼의 길이는 255 바이트를 초과할 수 없습니다. LOB 컬럼, DATALINK 컬럼, 또는 LOB나 DATALINK를 기초로 하는 구별 유형 컬럼은 컬럼의 길이 속성이 255 바이트 한계 내에 맞도록 충분히 작은 경우에도 색인의 부분으로 사용될 수 없습니다(SQLSTATE 42962). 구조화 유형 컬럼은 EXTEND USING 절도 지정될 경우에만 지정할 수 있습니다(SQLSTATE 42962). EXTEND USING 절을 지정할 경우, 하나의 컬럼만 지정할 수 있으며 컬럼의 유형이 LOB, DATALINK, LONG VARCHAR 또는 LONG VARGRAPHIC을 기초로 하는 구조화 유형이나 구별 유형이어야 합니다(SQLSTATE 42997).

ASC

색인 항목이 컬럼 값의 오름차순으로 보존됨을 지정합니다. 이것이 기본 설정입니다. ASC는 EXTEND USING으로 정의된 색인에 대해 지정할 수 없습니다(SQLSTATE 42601).

DESC

색인 항목이 컬럼 값의 내림차순으로 보존됨을 지정합니다. DESC는 EXTEND USING으로 정의된 색인에 대해 지정할 수 없습니다 (SQLSTATE 42601).

SPECIFICATION ONLY

이 명령문이 *별명(nickname)*에 의해 참조된 데이터 소스 테이블에 적용하는 색인 스펙을 작성하는 데 사용될 것임을 나타냅니다. *별명*이 지정되는 경우에는 SPECIFICATION ONLY가 지정되어야 합니다(SQLSTATE 42601). *테이블 이름*이 지정되는 경우에는 이것을 지정하면 안 됩니다(SQLSTATE 42601).

INCLUDE

이 키워드는 색인 키 컬럼 세트에 추가될 컬럼을 지정하는 절을 도입합니다. 고유성을 강화하기 위해 이 절에 포함된 컬럼이 사용되지는 않습니다. 이와 같은 포함된 컬럼들은 색인 전용 액세스를 통해서 일부 조회 성능을 향상시킬 수 있습니다. 컬럼들은 고유성을 강화시키는 데 사용되는 컬럼과 구별되어야 합니다(SQLSTATE 42711). 컬럼 수 및 길이 속성의 합계에 대한 한계는 고유 키 및 색인의 모든 컬럼에 적용됩니다.

컬럼 이름

색인에 포함되지만 고유 색인 키의 일부는 아닌 컬럼을 식별합니다. 고유한 색인 키 컬럼에 적용된 것과 동일한 규칙이 적용됩니다. 컬럼 이름 다음에 키워드 ASC 또는 DESC를 지정할 수 있으나 순서에는 영향을 미치지 않습니다.

INCLUDE는 EXTEND USING으로 정의된 색인에 대해, 또는 *별명(nickname)*이 지정된 경우에 지정할 수 없습니다(SQLSTATE 42601).

CLUSTER

색인이 테이블의 클러스터링 색인이 되도록 지정합니다. 클러스터링 색인의 클러스터 요소는 데이터가 관련 테이블로 삽입됨에 따라 이 색인의 키 값이 동일한 범위 내에 있는 행에 근접하도록 새로운 행을 물리적으로 삽입하여 동적으로 유지보수되거나 향상됩니다. 테이블 하나에 대해 하나의 클러스터링 색인만 존재할 수 있으므로, CLUSTER가 테이블에 있는 기존 색인 정의에 사용된 경우 지정할 수 없습니다. 부가 모드를 사용하기 위해 정의된 테이블에 클러스터 색인을 작성할 수 없습니다(SQLSTATE 428D8).

별명이 지정된 경우 CLUSTER는 허용되지 않습니다(42601).

EXTEND USING *색인 확장 이름*

이 색인을 관리하기 위해 사용되는 *색인 확장*에 이름을 부여합니다. 이 절을 지정할 경우, 하나의 *컬럼 이름*만 지정해야 하고 그 컬럼은 구조화 유형이나 구별 유형이어야 합니다(SQLSTATE 42997). *색인 표현식 이름*은 카탈로그에 설명된 색인 확장에 이름을 부여해야 합니다(SQLSTATE 42704). 구별 유형의 경우, 컬럼은 색인 확장에 있는 해당되는 소스 키 매개변수의 유형과 일치해야 합니다. 구조화 유형 컬럼의 경우, 해당되는 소스 키 매개변수 유형은 컬럼 유형과 같은 유형이거나 수퍼 유형이어야 합니다(SQLSTATE 428E0).

상수 표현식

색인 확장에 대한 필수 인수의 값을 식별합니다. 각 표현식은 길이나 정밀도, 그리고 스케일을 포함하여, 해당되는 색인 확장 매개변수의 정의된 데이터 유형과 정확하게 일치하는 데이터 유형을 갖는 상수 값이어야 합니다(SQLSTATE 428E0).

PCTFREE *정수*

색인 작성시, 빈 공간으로 남겨둘 색인 페이지 비율을 지정합니다. 페이지의 첫 번째 행은 제한 없이 추가됩니다. 색인 페이지에 추가 항목이 있는 경우, 적어도 *정수* 퍼센트의 빈 공간이 각 페이지에 남게 됩니다. *정수* 값의 범위는 0에서 99까지입니다. 그러나, 10보다 큰 값이 지정되는 경우, non-leaf 페이지에는 10퍼센트의 빈 공간만 남게 됩니다. 기본값은 10입니다.

별명이 지정된 경우 PCTFREE는 허용되지 않습니다(42601).

MINPCTUSED *정수*

색인이 온라인으로 재구성되는 지의 여부와 색인 리프 페이지에서 키가 삭제되어 페이지에서 사용되는 공간의 백분율이 *정수* 백분율 이하인 경우 이 페이지에 있는 나머지 키를 이웃하는 페이지의 키와 병합하려고 할 경우 색인 리프 페이지에 사용되는 공간의 최소 백분율에 대한 임계값을 나타냅니다. 이들 페이지 중 하나에 충분한 공간이 있으면, 병합이 수행되며 페이지들 중 하나가 삭제됩니다. *정수*값은 0에서 99까지입니다. 그러나, 성능상의 이유로 50 이하의 값이 권장됩니다.

별명이 지정된 경우 MINPCTUSED는 허용되지 않습니다(42601).

DISALLOW REVERSE SCANS

색인이 포워드 스캔이나 INDEX CREATE시에 정의된 순서로 색인을 스캔하는 것만 지원하도록 지정합니다. 이것은 기본값입니다.

*별명*이 지정된 경우 DISALLOW REVERSE SCANS는 허용되지 않습니다 (42601).

ALLOW REVERSE SCANS

색인이 포워드 스캔과 역 스캔을 모두 지원하도록 지정합니다. 즉, INDEX CREATE시에 정의된 순서와 그 반대 순서를 지원합니다.

*별명*이 지정된 경우 ALLOW REVERSE SCANS는 허용되지 않습니다 (42601).

규칙

- CREATE INDEX문은 기존 색인과 일치하는 색인을 작성하려고 하는 경우에 실패합니다(SQLSTATE 01550). 다음과 같은 경우 두 개의 색인 설명은 중복된 것으로 간주됩니다.
  - 컬럼 세트(키와 include 컬럼 모두)와 색인에 있는 순서가 기존의 색인 순서와 동일한 경우
  - 순서 속성이 동일한 경우
  - 이전의 기존 색인과 작성중인 색인이 둘 다 고유하지 않거나 이전의 기존 색인이 교유할 경우
  - 이전의 기존 색인과 작성중인 색인이 고유할 경우, 작성중인 색인의 키 컬럼은 이전의 기존 색인의 키 컬럼과 동일하거나 상위 세트입니다.

주

- 명명된 테이블에 데이터가 이미 포함되어 있는 경우, CREATE INDEX는 이 데이터에 대한 색인 항목을 작성합니다. 테이블에 아직 데이터가 포함되어 있지 않으면, CREATE INDEX는 색인의 설명을 작성하고, 색인 항목은 데이터가 그 테이블에 삽입될 때 작성됩니다.
- 일단 색인이 작성되어 데이터가 테이블에 로드되면, RUNSTATS 명령을 실행하는 것이 좋습니다(RUNSTATS에 대해서는 *Command Reference*에서 정보를 참조하십시오.). RUNSTATS 명령은 데이터베이스 테이블, 컬럼 및 색인에 수

집된 통계를 갱신합니다. 이 통계는 테이블에 대한 최적의 액세스 경로를 판별할 때 사용됩니다. RUNSTATS 명령을 실행하여, 데이터베이스 관리 프로그램이 새로운 색인의 특성을 판별할 수 있습니다.

- 아직 존재하지 않는 스키마 이름을 사용하여 색인을 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 최적화 알고리즘은 실제 색인 작성에 앞서 색인을 권장할 수 있습니다. 1145 페이지의 『SET CURRENT EXPLAIN MODE』에서 자세한 내용을 참조하십시오.
- 색인을 가지고 있는 데이터 소스에 대해 색인 스펙이 정의되고 있다면, 색인 스펙의 이름이 색인의 이름과 일치할 필요가 없습니다.
- 최적화 알고리즘은 색인 스펙을 사용하여 스펙이 적용하는 데이터 소스 테이블에 대한 액세스를 향상시킵니다.
- 색인 스펙에 대한 57 페이지의 『색인 스펙』에서 자세한 정보를 참조하십시오.

예

예 1: PROJECT 테이블에서 UNIQUE_NAM이라고 하는 색인을 작성합니다. 색인의 목적은 프로젝트 이름(PROJNAME)에 대해 같은 값을 갖고 있는 항목이 테이블에 두 개가 되지 않도록 하는 것입니다. 색인 항목은 오름차순으로 됩니다.

```
CREATE UNIQUE INDEX UNIQUE_NAM
 ON PROJECT(PROJNAME)
```

예 2: EMPLOYEE 테이블에서 JOB_BY_DPT라고 하는 색인을 작성합니다. 각 부서(WORKDEPT) 내에서 직책(JOB)별로 오름차순으로 색인 항목을 배열합니다.

```
CREATE INDEX JOB_BY_DPT
 ON EMPLOYEE (WORKDEPT, JOB)
```

예 3: 별명 EMPLOYEE는 CURRENT_EMP라고 하는 데이터 소스 테이블을 참조합니다. 이 별명이 작성된 후, CURRENT_EMP에서 색인이 정의되었습니다. 색인 키에 대해 선택된 컬럼은 WORKDEBT와 JOB였습니다. 이 색인을 기술하는

색인 스펙을 작성하십시오. 이 스펙을 통해, 최적화 알고리즘은 색인이 존재하는지와 그 키가 무엇인지를 알 것입니다. 이 정보로 최적화 알고리즘은 테이블 액세스 전략을 향상시킬 수 있습니다.

```
CREATE UNIQUE INDEX JOB_BY_DEPT
ON EMPLOYEE (WORKDEPT, JOB)
   SPECIFICATION ONLY
```

예 4: 구조화 유형 컬럼 위치에 SPATIAL_INDEX라고 하는 확장된 색인 유형을 작성합니다. 색인 확장 GRID_EXTENSION에 있는 설명은 PATIAL_INDEX의 유지보수에 사용됩니다. 이 리터럴은 색인 격자 *크기*를 작성하기 위해 GRID_EXTENSION에 제공됩니다. 색인 확장 정의에 대해서는 749 페이지의 『CREATE INDEX EXTENSION』에서 자세한 내용을 참조하십시오.

```
CREATE INDEX SPATIAL_INDEX ON CUSTOMER (LOCATION)
     EXTEND USING (GRID_EXTENSION (x'000100100010001000400010'))
```

CREATE INDEX EXTENSION

CREATE INDEX EXTENSION문은 구조화 유형이나 구별 유형 컬럼을 가지고 있는 테이블에서 색인과 함께 사용하기 위해 확장 오브젝트를 작성합니다.

호출

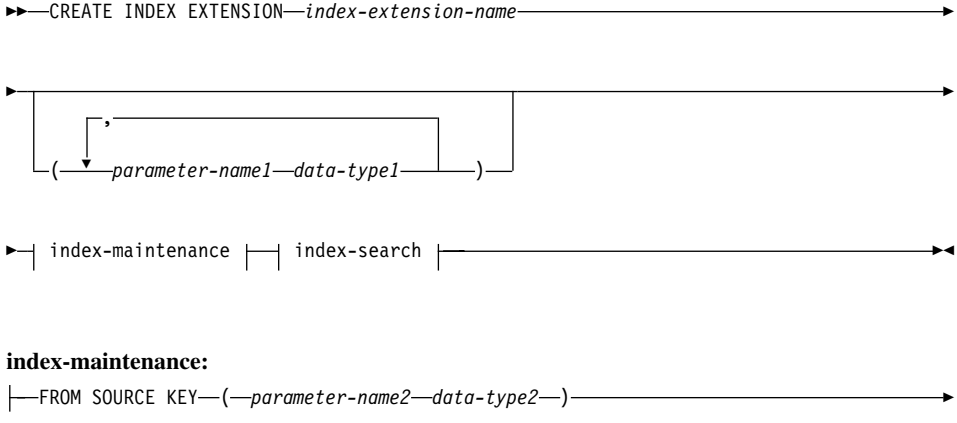
이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 통해 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

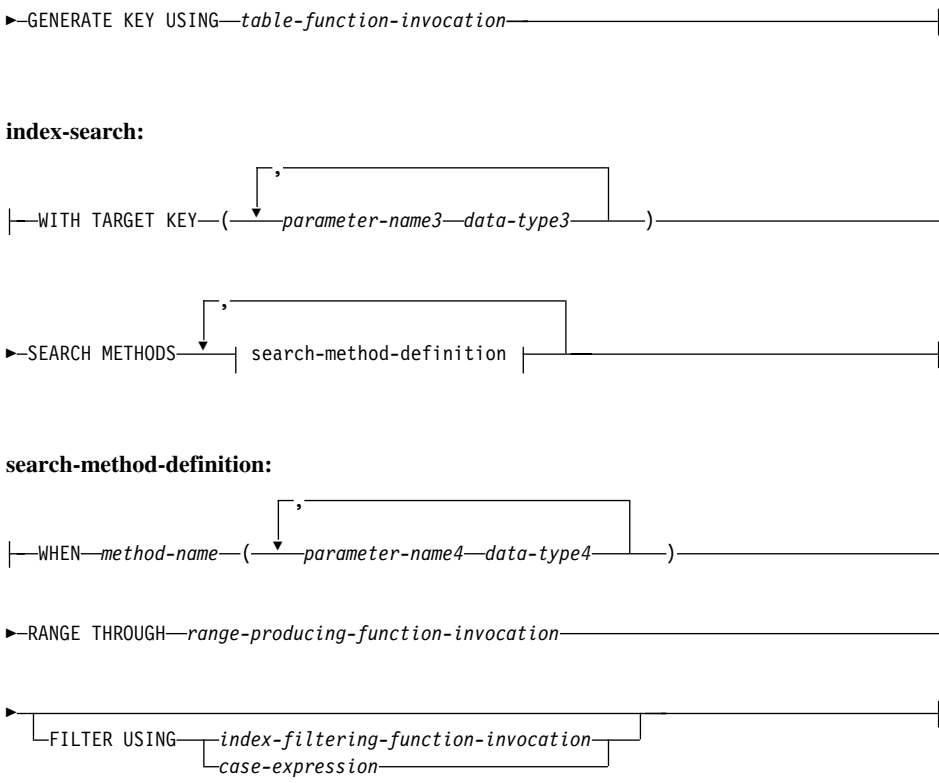
명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 데이터베이스에 대한 IMPLICIT_SCHEMA 권한(색인 확장의 스키마 이름이 기존 스키마를 참조하지 않을 경우)
- 스키마에 대한 CREATEIN 특권(색인 확장의 스키마 이름이 기존 스키마를 참조할 경우)

구문



index-maintenance:



설명

색인 확장 이름

색인 확장에 이름을 부여합니다. 내재적 또는 명시적 규정자를 포함하는 이름은 카탈로그에 설명된 색인 확장을 식별하지 말아야 합니다. 두 부분 *색인 확장 이름*을 지정하면, 스키마 이름은 "SYS"로 시작할 수 없습니다. 그렇지 않으면, 오류(SQLSTATE 42939)가 발생합니다.

매개변수 이름1

CREATE INDEX 시간에 색인 확장의 실제 동작을 정의하기 위해 색인 확장에 전달되는 매개변수를 식별합니다. 색인 확장에 전달되는 매개변수는 *인스턴스 매개변수*라고 합니다. 이는 그 값이 색인 확장의 새로운 인스턴스를 정의하기 때문입니다.

*매개변수 이름1*은 색인 확장의 정의 내에서 고유해야 합니다. 매개변수 개수는 90개를 초과할 수 없습니다. 이 한계를 초과하면, 오류가 발생합니다(SQLSTATE 54023).

데이터 유형1

각 매개변수의 데이터 유형을 지정합니다. 색인 확장에서 받기를 기대하는 각 매개변수에 대해 목록에서 한 항목만 지정해야 합니다. 지정할 수 있는 유일한 SQL 데이터 유형은 VARCHAR, INTEGER, DECIMAL, DOUBLE 또는 VARGRAPHIC과 같이, 상수로 사용될 수 있는 데이터 유형입니다(SQLSTATE 429B5). 상수에 대해서는 133 페이지의 『상수』에서 자세한 내용을 참조하십시오. CREATE INDEX에서 색인 확장에서 받는 매개변수 값은 길이, 정밀도 및 스케일을 포함하여, 정확하게 *데이터 유형1*과 일치해야 합니다(SQLSTATE 428E0).

색인 유지보수

구조화 또는 구별 유형 컬럼의 색인 키가 어떻게 유지보수되는 지를 지정합니다. 색인 유지보수는 소스 컬럼을 목표 키로 변환하는 프로세스입니다. 변환 프로세스는 이전에 데이터베이스에서 정의된 테이블 함수를 사용하여 정의됩니다.

FROM SOURCE KEY (*매개변수 이름2 데이터 유형2*)

색인 확장에서 지원되는 소스 키 컬럼에 대한 구조화 유형 또는 구별 유형을 지정합니다.

매개변수 이름2

소스 키 컬럼과 연관되는 매개변수를 식별합니다. 소스 키 컬럼은 데이터 유형이 *데이터 유형2*와 같은 색인 키 컬럼(CREATE INDEX문에 정의된)입니다.

데이터 유형2

*매개변수 이름2*에 대한 데이터 유형을 지정합니다. *데이터 유형2*는 LOB, DATALINK, LONG VARCHAR 또는 LONG VARGRAPHIC에서 전래되지 않은 구별 유형이나 사용자 정의 구조화 유형이어야 합니다(SQLSTATE 42997). 색인 확장이 CREATE INDEX에서 색인과 연관될 경우, 그 색인 키 컬럼의 데이터 유형은 다음과 같아야 합니다.

- 구별 유형일 경우 정확하게 *데이터 유형2*와 일치해야 합니다.

- 구조화 유형일 경우 *데이터 유형2*와 같은 유형이거나 부속 유형이어야 합니다.

그렇지 않으면, 오류가 발생합니다(SQLSTATE 428E0).

GENERATE KEY USING *테이블 함수 호출*

사용자 정의 테이블 함수를 사용하여 색인 키가 생성되는 방법을 지정합니다. 단일 소스 키 데이터 값에 대해 여러 개의 색인 항목이 생성될 수도 있습니다. 색인 항목은 단일 소스 키 데이터 값에서 중복될 수 없습니다(SQLSTATE 22526). 함수는 *매개변수 이름1*, *매개변수 이름2* 또는 상수를 인수로 사용할 수 있습니다. *매개변수 이름2*의 데이터 유형이 구조화 데이터 유형일 경우, 그 구조화 유형의 observer 메소드만 해당되는 인수에서 사용될 수 있습니다(SQLSTATE 428E3). GENERATE KEY 함수의 출력은 TARGET KEY 스펙에서 지정해야 합니다. 함수 출력은 FILTER USING 절에 지정된 색인 필터링 함수에 대한 입력으로 사용할 수도 있습니다.

테이블 함수 호출에서 사용되는 함수는 다음과 같아야 합니다.

1. 테이블 함수로 분석되어야 합니다(SQLSTATE 428E4).
2. LANGUAGE SQL로 정의하지 않아야 합니다(SQLSTATE 428E4).
3. NOT DETERMINISTIC(SQLSTATE 428E4) 또는 EXTERNAL ACTION(SQLSTATE 428E4)으로 정의하지 않아야 합니다.
4. 시스템에서 생성되는 observer 메소드를 제외하고, 매개변수의 데이터 유형에서 구조화 데이터 유형으로 LOB, DATALINK, LONG VARCHAR 또는 LONG VARGRAPHIC을 가지고 있지 않아야 합니다(SQLSTATE 428E3).
5. 부속 조회를 포함하고 있지 않아야 합니다(SQLSTATE 428E3).
6. EXTEND USING 절 없이 정의된 색인 컬럼의 데이터 유형에 대한 제한사항을 따르는 데이터 유형의 컬럼을 리턴해야 합니다.

인수가 다른 조작이나 루틴을 호출할 경우, 이것은 observer 메소드여야 합니다(SQLSTATE 428E3).

색인 탐색

검색 인수의 맵핑을 검색 범위에 제공하여 검색이 수행되는 방법을 지정합니다.

WITH TARGET KEY

GENERATE KEY USING 절에 지정된 키 생성 함수의 출력인 목표 키 매개변수를 지정합니다.

매개변수 이름3

주어진 목표 키와 연관되는 매개변수를 지정합니다. *매개변수 이름3*은 GENERATE KEY USING 절의 테이블 함수에 지정된 RETURNS 테이블의 컬럼에 해당됩니다. 지정된 매개변수 개수는 테이블 함수에서 리턴되는 컬럼 수와 일치해야 합니다(SQLSTATE 428E2).

데이터 유형3

해당되는 각 *매개변수 이름3*에 대한 데이터 유형을 지정합니다. *데이터 유형3*은 길이, 정밀도 및 유형을 포함하여, GENERATE KEY USING 절의 테이블 함수에 지정된 대로, RETURNS 테이블의 해당되는 출력 데이터 유형과 정확히 일치해야 합니다(SQLSTATE 428E2).

SEARCH METHODS

색인에 대해 정의된 검색 메소드를 도입합니다.

탐색 메소드 정의

색인 검색에 대한 메소드 세부사항을 지정합니다. 이것은 메소드 이름, 검색 인수, 범위 생성 함수, 그리고 선택적 색인 필터 함수로 구성됩니다.

WHEN *메소드 이름*

탐색 메소드의 이름. 이것은 색인 노출 규칙에 지정된 메소드 이름과 관련되는 SQL 식별자입니다(사용자 정의 함수의 PREDICATES 절에 있습니다). *검색 메소드 이름*은 검색 메소드 정의에서 단 하나의 WHEN 절에서만 참조될 수 있습니다(SQLSTATE 42713).

매개변수 이름4

탐색 인수의 매개변수를 식별합니다. 이 이름들은 RANGE THROUGH 및 FILTER USING 절에서 사용됩니다.

데이터 유형4

검색 매개변수와 연관되는 데이터 유형.

RANGE THROUGH *범위 생성 함수 호출*

검색 범위를 생성하는 외부 테이블 함수를 지정합니다. 이 함수는 인수로 *매개변수 이름1*, *매개변수 이름4* 또는 상수를 사용하여 검색 범위 세트를 리턴합니다.

범위 생성 함수 호출에서 사용되는 테이블 함수는 다음과 같아야 합니다.

1. 테이블 함수로 분석되어야 합니다(SQLSTATE 428E4).
2. 부속 조회(SQLSTATE 428E3)나 SQL 함수(SQLSTATE 428E4)를 포함하고 있지 않아야 합니다.
3. LANGUAGE SQL로 정의하지 않아야 합니다(SQLSTATE 428E4).
4. NOT DETERMINISTIC 또는 EXTERNAL ACTION으로 정의하지 않아야 합니다(SQLSTATE 428E4).
5. 이 함수 결과의 번호와 유형은 다음과 같이 GENERATE KEY USING 절에 지정된 테이블 함수의 결과와 관련되어야 합니다(SQLSTATE 428E1).
   - 키 변환 함수에 의해 리턴되는 컬럼 수의 두 배까지 리턴합니다.
   - 짝수 개수의 컬럼이 있습니다. 리턴 컬럼의 처음 반은 범위의 시작(시작 키 값)을 정의하고, 리턴 컬럼의 두 번째 반은 범위의 끝(중지 키 값)을 정의합니다.
   - 각 시작 키 컬럼에는 같은 유형의 해당되는 중지 키 컬럼이 있습니다.
   - 각 시작 키 컬럼의 유형은 해당되는 키 변환 함수 컬럼의 유형과 같습니다.

더 명확하게, $_1$:t_1, ..., a_n:t_n을 함수 결과 컬럼과 키 변환 함수의 데이터 유형이라고 합시다. *범위 생성 함수* 호출의 함수 결과는 b_1:t_1, ..., b_m:t_m, c_1:t_1, ..., c_m:t_m이어야 합니다. 여기서 m <= n과 "b" 컬럼은 시작 키 컬럼이고 "c" 컬럼은 중지 키 컬럼입니다.

범위 생성 함수 호출이 시작 및 중지 키 값으로 널(NULL) 값을 리턴할 경우, 의미론은 정의되지 않습니다.

FILTER USING

외부 함수나 CASE 표현식의 스펙이 범위 생성 함수 적용후 리턴된 색인 항목을 필터링하는 데 사용되도록 합니다.

색인 필터링 함수 호출

색인 항목 필터링에 사용될 외부 함수를 지정합니다. 이 함수는 *매개변수 이름1*, *매개변수 이름3*, *매개변수 이름4* 또는 상수를 인수로 사용하고 (SQLSTATE 42703), 정수를 리턴합니다(SQLSTATE 428E4). 리턴된 값이 1일 경우, 색인 항목에 해당되는 행은 테이블에서 검색됩니다. 그렇지 않으면, 색인 항목은 추가 처리에 고려되지 않습니다.

지정하지 않으면, 색인 필터링이 수행되지 않습니다.

색인 필터링 함수 호출에 사용되는 함수는 다음과 같아야 합니다.

1. LANGUAGE SQL로 정의하지 않아야 합니다(SQLSTATE 429B4).
2. NOT DETERMINISTIC 또는 EXTERNAL ACTION으로 정의하지 않아야 합니다(SQLSTATE 42845).
3. 매개변수 중 어느 하나의 데이터 유형으로 구조화 데이터 유형을 가지고 있지 않아야 합니다(SQLSTATE 428E3)
4. 부속 조회를 포함하고 있지 않아야 합니다(SQLSTATE 428E3).

인수가 다른 함수나 메소드를 호출할 경우 중첩된 함수나 메소드에 대해 위의 네 가지 규칙도 시행됩니다. 그러나, 시스템에서 생성되는 observer 메소드는 인수가 내장 데이터 유형을 야기하면 필터 함수(또는 인수로 사용되는 함수나 메소드)에 대한 인수로 허용됩니다.

CASE-표현식

색인 항목 필터링에 대한 CASE 표현식을 지정합니다. *매개변수 이름1*, *매개변수 이름3*, *매개변수 이름4* 또는 상수(SQLSTATE 42703)를 *searched-when-clause* 및 *simple-when-clause*에 사용할 수 있습니다. FILTER USING *색인 필터링 함수 호출*에 규칙이 지정된 외부 함수를 *결과 표현식*으로 사용할 수 있습니다. *CASE 표현식*에서 참조되는 함수는 *색인 필터링 함수 호출* 아래에 나열된 네 가지 규칙도 따라야 합니다. 부속 조회는 *CASE 표현식* 어디에서도 사용할 수 없습니다(SQLSTATE 428E4).

CASE 표현식은 정수를 리턴해야 합니다(SQLSTATE 428E4). *결과 표현식*에서 리턴 값 1은 색인 항목이 보존됨을 의미하고, 그렇지 않으면 그 색인 항목은 버려집니다.

주

- 아직 존재하지 않는 스키마 이름을 사용하여 색인 표현식을 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.

예

예 1: 다음은 *gridEntry*라고 하는 테이블 함수에서 구조화 유형 SHAPE 컬럼을 사용하여 7개의 색인 목표 키를 생성하는 *grid_extension*이라고 하는 색인 표현식을 작성합니다. 이 색인 확장자는 또한 검색 인수가 제공될 때 검색 범위를 생성하기 위해 두 개의 색인 검색 메소드를 제공합니다.

```
CREATE INDEX EXTENSION GRID_EXTENSION (LEVELS VARCHAR(20) FOR BIT DATA)
   FROM SOURCE KEY (SHAPECOL SHAPE)
   GENERATE KEY USING GRIDENTRY(SHAPECOL..MBR..XMIN,
                                SHAPECOL..MBR..YMIN,
                                SHAPECOL..MBR..XMAX,
                                SHAPECOL..MBR..YMAX,
                                LEVELS)
   WITH TARGET KEY (LEVEL INT, GX INT, GY INT,
                    XMIN INT, YMIN INT, XMAX INT, YMAX INT)
   SEARCH METHODS
   WHEN SEARCHFIRSTBYSECOND (SEARCHARG SHAPE)
   RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
                                SEARCHARG..MBR..YMIN,
                                SEARCHARG..MBR..XMAX,
                                SEARCHARG..MBR..YMAX,
                                LEVELS)
   FILTER USING
      CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR SEARCHARG..MBR..YMAX < YMIN) THEN 0
      ELSE CHECKDUPLICATE(LEVEL, GX, GY,
                                   XMIN, YMIN, XMAX, YMAX,
                                   SEARCHARG..MBR..XMIN,
                                   SEARCHARG..MBR..YMIN,
                                   SEARCHARG..MBR..XMAX,
                                   SEARCHARG..MBR..YMAX,
                                   LEVELS)
      END
   WHEN SEARCHSECONDBYFIRST (SEARCHARG SHAPE)
   RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
                                SEARCHARG..MBR..YMIN,
                                SEARCHARG..MBR..XMAX,
                                SEARCHARG..MBR..YMAX,
                                LEVELS)
   FILTER USING
      CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR SEARCHARG..MBR..YMAX < YMIN) THEN 0
      ELSE MBROVERLAP(XMIN, YMIN, XMAX, YMAX,
                                SEARCHARG..MBR..XMIN,
```

```
                                SEARCHARG..MBR..YMIN,
                                SEARCHARG..MBR..XMAX,
                                SEARCHARG..MBR..YMAX)
END
```

CREATE METHOD

이 명령문은 이미 사용자가 정의하는 구조화 유형의 일부인 메소드 스펙과 메소드 내용을 연관시키는 데 사용됩니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

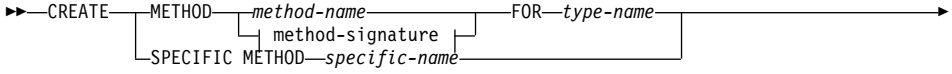
명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

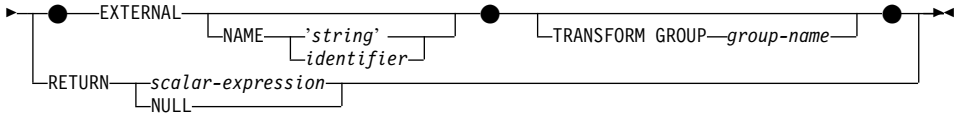
- SYSADM 또는 DBADM 권한
- CREATE METHOD에서 참조된 구조화 유형의 스키마에 대한 CREATEIN 특권.
- CREATE METHOD문에서 참조된 구조화 유형의 DEFINER.

명령문의 권한 부여 ID가 SYSADM 또는 DBADM 권한을 가지고 있지 않고, 메소드가 RETURN문에서 테이블이나 뷰를 식별할 경우, 명령문의 권한 부여 ID가 보유하는 특권에는(고려하는 그룹 특권 없이) 식별된 각 테이블과 뷰에 대한 SELECT WITH GRANT OPTION이 포함되어야 합니다.

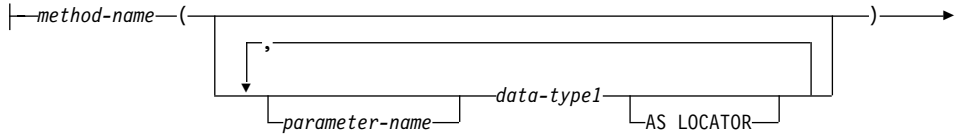
권한 부여 ID가 조작을 수행할 권한을 가지고 있지 않으면, 오류가 발생합니다 (SQLSTATE 42502).

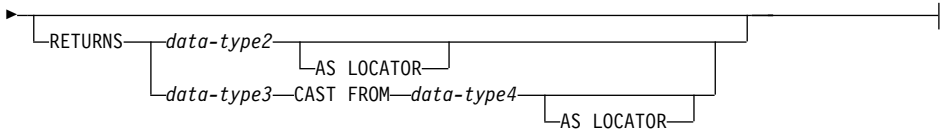
구문





method-signature:





설명

METHOD

사용자가 정의하는 구조화 유형과 연관되는 기존의 메소드 스펙을 식별합니다. 메소드 스펙은 다음 수단 중 한 가지 방법으로 식별될 수 있습니다.

메소드 이름

메소드 본문이 정의되는 메소드 스펙에 이름을 지정합니다. 내재된 스키마는 주제 유형이 스키마입니다(*유형 이름*). 이 *메소드 이름*을 가지고 있는 *유형 이름*에 대해 하나의 메소드 스펙만 있어야 합니다(SQLSTATE 42725).

메소드 시그니처

정의할 메소드를 고유하게 식별하는 메소드 시그니처를 제공합니다. 메소드 시그니처는 CREATE TYPE 또는 ALTER TYPE 명령문에 제공된 메소드 스펙과 일치해야 합니다(SQLSTATE 42883).

메소드 이름

메소드 본문이 정의되는 메소드 스펙에 이름을 지정합니다. 내재된 스키마는 주제 유형이 스키마입니다(*유형 이름*).

매개변수 이름

매개변수 이름을 식별합니다. 메소드 시그니처에서 매개변수 이름이 제공되면, 그 이름은 일치하는 메소드 스펙의 해당되는 부분과 정확하게 같아야 합니다. 매개변수 이름은 문서화를 위해서만 이 명령문에서 지원됩니다.

데이터 유형1

각 매개변수의 데이터 유형을 지정합니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다.

RETURNS

이 절은 메소드의 출력을 식별합니다. 메소드 시그니처에서 RETURNS 절이 제공되면, 이는 CREATE TYPE에서 일치하는 메소드 스펙의 해당되는 부분과 정확하게 같아야 합니다. RETURNS 절은 문서화를 위해서만 이 명령문에서 지원됩니다.

데이터 유형2

출력의 데이터 유형을 지정합니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 실제 값 대신 LOB 위치 지정자가 메소드에 의해 리턴됨을 나타냅니다.

데이터 유형3 **CAST FROM** *데이터 유형4*

이 형태의 RETURNS절은 함수 코드에 의해 리턴된 데이터 유형에서 호출 명령문으로 서로 다른 데이터 유형을 리턴하는 데 사용됩니다.

AS LOCATOR

LOB 유형이나 LOB 유형을 기초로 하는 구별 유형의 경우, AS LOCATOR 절은 실제 값 대신 메소드로부터 LOB 위치 지정자가 리턴됨을 나타내기 위해 사용할 수 있습니다.

FOR *유형 이름*

지정된 메소드가 연관될 유형의 이름을 지정합니다. 이 이름은 카탈로그에 이미 설명된 유형을 식별해야 합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

SPECIFIC METHOD *고유 이름*

CREATE TYPE 시에 지정하였거나 기본값으로 설정된 특정 이름을 사용하여 특정 메소드를 식별합니다. 특정 이름은 명명된 스키마 또는 내재된 스키마에서 메소드 스펙을 식별해야 합니다. 그렇지 않으면, 오류가 발생합니다 (SQLSTATE 42704).

EXTERNAL

이 절은 CREATE METHOD문이 외부 프로그래밍 언어로 작성된 코드를 기초로 문서화된 연계 규칙과 인터페이스에 따라 새 함수를 등록하는 데 사용됨을 나타냅니다. CREATE TYPE에서 일치하는 메소드 스펙은 SQL 이외의 LANGUAGE를 지정해야 합니다. 메소드가 호출될 경우, 메소드의 주제는 내재된 첫번째 매개변수로서 구현에 전달됩니다.

NAME절이 지정되지 않았을 경우, "NAME *메소드 이름*"이 사용됩니다.

NAME

이 절은 정의되는 메소드를 구현하는 사용자 작성 코드의 이름을 식별합니다.

'*문자열***'**

'문자열' 옵션은 최대 254 문자로 된 문자열 상수입니다. 문자열에 사용되는 형식은 지정되는 LANGUAGE에 따라 다릅니다. 특정 언어 규칙에 대해서는 655 페이지의 『CREATE FUNCTION(외부 스칼라)』에서 자세한 내용을 참조하십시오.

식별자

지원된 이 식별자는 SQL 식별자입니다. SQL 식별자는 문자열에서 library-id로 사용됩니다. 분리 식별자가 아닐 경우, 식별자는 대문자로 겹쳐집니다. 식별자에 스키마 이름이 규정된 경우, 스키마 이름 부

분은 무시됩니다. 이 NAME 양식은 LANGUAGE C에서만 사용할 수 있습니다(CREATE TYPE의 메소드 스펙에서 정의된 대로).

TRANSFORM GROUP *그룹 이름*

메소드를 호출할 때 사용자가 정의하는 구조화 유형변환에 대해 사용되는 변환 그룹을 나타냅니다. 변환은 메소드 정의에 사용자가 정의하는 구조화 유형이 포함되므로 반드시 필요합니다.

변환 그룹 이름을 지정하는 것이 아주 좋습니다. 이 절을 지정하지 않을 경우, 사용되는 기본 그룹 이름은 DB2_FUNCTION입니다. 지정된 (또는 기본값인) 그룹 이름이 참조된 구조화 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42741). 마찬가지로, 필수 FROM SQL 또는 TO SQL 변환 함수가 제공된 그룹 이름과 구조화 유형에 대해 정의되지 않은 경우, 오류가 발생합니다(SQLSTATE 42744).

RETURN *스칼라 표현식* **또는 NULL**

RETURN 명령문은 메소드에 의해 리턴되는 값을 지정하는 SQL 제어 명령문입니다.

스칼라 표현식

CREATE TYPE의 메소드 스펙이 LANGUAGE SQL을 지정할 경우 메소드의 내용을 지정하는 표현식. 매개변수 이름은 표현식에서 참조될 수 있습니다. 메소드의 주제는 내재된 첫번째 매개변수인 SELF 양식으로 메소드 구현에 전달됩니다. 표현식의 결과 데이터 유형은 CREATE TYPE에서 메소드 스펙의 RETURNS 절에 정의된 데이터 유형에 지정할 수 있어야 합니다(저장 지정 규칙을 사용하여)(SQLSTATE 42866).

표현식은 메소드 스펙의 다음 부분을 따라야 합니다.

- DETERMINISTIC 또는 NOT DETERMINISTIC (SQLSTATE 428C2)
- EXTERNAL ACTION 또는 NO EXTERNAL ACTION (SQLSTATE 428C2)
- CONTAINS SQL 또는 READS SQL DATA (SQLSTATE 42985)

NULL

함수가 널(NULL) 값을 리턴함을 지정합니다. 널(NULL) 값은 CREATE TYPE문에서 작성된 메소드 스펙의 RETURNS 절에 정의된 데이터 유형의 값입니다.

규칙

메소드 스펙은 CREATE METHOD를 사용하기 전에 CREATE TYPE이나 ALTER TYPE문을 사용하여 이전에 정의되어 있어야 합니다(SQLSTATE 42723).

예

예 1:

```
CREATE METHOD BONUS (RATE DOUBLE)
   FOR EMP
   RETURN SELF..SALARY * RATE
```

예 2:

```
CREATE METHOD SAMEZIP (addr address_t)
   RETURNS INTEGER
   FOR address_t
   RETURN
      (CASE
         WHEN (self..zip = addr..zip)
            THEN 1
         ELSE 0
      END)
```

예 3:

```
CREATE METHOD DISTANCE (address_t)
   FOR address_t
   EXTERNAL NAME 'addresslib!distance'
   TRANSFORM GROUP func_group
```

CREATE NICKNAME

CREATE NICKNAME문은 데이터 소스 테이블이나 뷰에 대한 별명(nickname)을 작성합니다.

호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

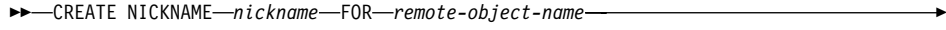
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 별명의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 연합 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 별명(nickname)의 스키마 이름이 존재하는 경우, 스키마에 대한 CREATEIN 특권.

또한, 데이터 소스에 있는 사용자의 권한 부여 ID가 데이터 소스 카탈로그로부터 별명이 작성되고 있는 해당 테이블이나 뷰에 대한 메타데이터를 선택할 특권을 보유해야 합니다.

구문



설명

별명(nickname)

*원격 오브젝트 이름*에 의해 참조되는 테이블이나 뷰에 대한 연합 서버의 식별자에 이름을 부여하십시오. 내재적 또는 명시적 규정자를 포함하여 별명

(nickname)은 카탈로그에 기술된 테이블, 뷰, 별명(alias) 또는 별명(nickname)을 식별해선 안 됩니다. 스키마 이름은 SYS로 시작하면 안 됩니다(SQLSTATE 42939).

원격 오브젝트 이름

이 포맷으로 세 부분으로 된 식별자에 이름을 부여합니다.

데이터 소스 이름.원격 스키마 이름.원격 테이블 이름

설명

데이터 소스 이름

작성되고 있는 별명(nickname에 대한 테이블이나 뷰를 포함하는 데이터 소스에 이름을 부여합니다. *데이터 소스 이름*은 CREATE SERVER문에서 데이터 소스에 지정된 것과 같은 이름입니다.

원격 스키마 이름

테이블이나 뷰가 속하는 스키마에 이름을 부여합니다.

원격 테이블 이름

다음 식별자 중 하나에 이름을 부여합니다.

- DB2 계열 테이블이나 뷰의 이름이나 별명(alias)
- Oracle 테이블이나 뷰의 이름
- *테이블 이름*은 선언된 임시 테이블이 될 수 없습니다(SQLSTATE 42995).

주

- 별명이 참조하는 테이블이나 뷰가 이미 *원격 오브젝트 이름*의 첫번째 규정자에 의해 표시된 데이터 소스에 존재해야 합니다.
- 연합 서버는 LONG VARCHAR, LONG VARGRAPHIC, DATALINK, 대형 오브젝트(LOB) 유형 및 사용자 정의 유형과 같은 DB2 데이터 유형에 해당되는 데이터 소스 데이터 유형은 지원하지 않습니다. 데이터 소스 테이블이나 뷰에 대해 별명이 정의될 때에는, 지원되는 데이터 유형이 정의될 테이블이나 뷰에 있는 컬럼만 연합 서버로부터 조회될 수 있습니다. 지원되지 않는 데이터 유형을 가진 컬럼이 있는 테이블이나 뷰에 대해 CREATE NICKNAME문이 실행되면 오류가 발행됩니다.

- 데이터 소스간에 데이터 유형이 호환되지 않을 수도 있으므로, 연합 서버는 필요에 따라 지역으로 원격 카탈로그 데이터를 보관하기 위해 약간의 조정을 취합니다. *응용프로그램 개발 안내서*에서 자세한 내용을 참조하십시오.
- DB2 색인 이름의 최대 허용 길이는 18문자입니다. 그 이름이 이 길이를 초과하는 색인을 가지는 테이블에 대해 별명이 작성되고 있다면, 전체 이름이 카탈로그화되지 않습니다. 대신, DB2는 이를 18문자로 절단합니다. 이들 문자로 형성된 문자열이 색인이 속하는 스키마 내에서 고유하지 않으면, DB2가 마지막 문자를 0으로 대체하여 이를 고유하게 만들려고 시도합니다. 결과가 여전히 고유하지 않으면, DB2가 마지막 문자를 1로 바꿉니다. DB2는 필요하다면 2에서 9까지의 문자로 고유한 이름이 생성될 때까지 이 프로세스를 반복합니다. 데이터 소스의 색인은 ABCDEFGHIJKLMNOPQRSTUVWXYZ라 명명됩니다. 이름 ABCDEFGHIJKLMNOPQR 및 ABCDEFGHIJKLMNOPQ0은 이미 이 색인이 속하는 스키마에 존재합니다. 새로운 이름은 18문자를 넘으며, 따라서 DB2가 이를 ABCDEFGHIJKLMNOPQR로 절단합니다. 이 이름이 이미 스키마에 존재하기 때문에, DB2는 절단된 버전을 ABCDEFGHIJKLMNOPQ0으로 변경합니다. 이 나중 이름 역시 존재하기 때문에, DB2는 절단된 버전을 ABCDEFGHIJKLMNOPQ1로 변경합니다. 이 이름은 스키마에 아직 없으며, 따라서 DB2는 이제 이것을 새로운 이름으로 받아들입니다.
- 테이블이나 뷰에 대한 별명(nickname)이 작성될 때, DB2는 테이블이나 뷰 컬럼의 이름을 카탈로그에 보관합니다. 이름이 DB2 컬럼 이름에 대해 허용되는 최대 길이(30자)를 초과할 경우, DB2는 이 길이로 이름을 절단합니다. 절단된 버전이 테이블이나 뷰의 컬럼에 있는 다른 이름들 사이에서 고유하지 않으면, DB2는 이전 단락에서 설명된 프로시듀어에 따라 이름을 고유하기 합니다.

예

예 1: HEDGES라 불리는 스키마에 있는 뷰 DEPARTMENT에 대한 별명을 작성하십시오. 이 뷰는 OS390A라고 하는 OS/390용 DB2 Universal Database 데이터 소스에 보관됩니다.

```
CREATE NICKNAME DEPT FOR OS390A.HEDGES.DEPARTMENT
```

예 2: 예 1에서 작성된 별명에 대한 뷰로부터 모든 레코드를 선택하십시오. 뷰는 별명으로 참조되어야 합니다. 통과 세션에서만 자신의 이름으로 참조될 수 있습니다.

```
SELECT * FROM OS390A.HEDGES.DEPARTMENT   유효하지 않음
SELECT * FROM DEPT                       별명 DEPT가 작성된 후 유효함
```

CREATE NODEGROUP

CREATE NODEGROUP문은 데이터베이스내에 새로운 노드 그룹을 작성하고 파티션 또는 노드를 노드 그룹에 할당한 다음 노드 그룹 정의를 카탈로그에 기록합니다.

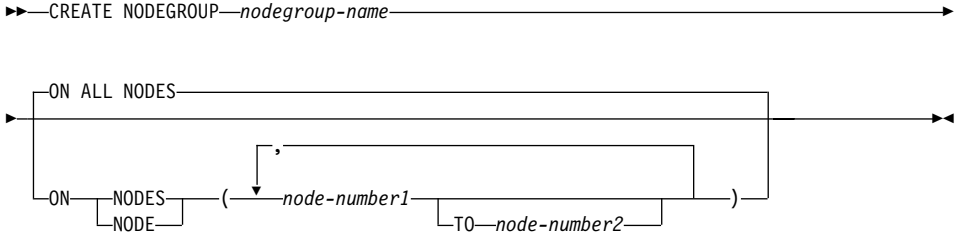
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비할 수 있는 실행 가능 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에는 SYSCTRL 또는 SYSADM 권한이 있어야 합니다.

구문



설명

노드 그룹 이름

노드 그룹의 이름을 지정합니다. 이 이름은 한 부분의 이름입니다. 이것은 SQL 식별자(일반 또는 분리 식별자)입니다. 노드 그룹 이름은 카탈로그에 이미 존재하는 노드 그룹을 나타내면 안됩니다(SQLSTATE 42710). 노드 그룹 이름은 문자 "SYS" 또는 "IBM"으로 시작하면 안됩니다(SQLSTATE 42939).

ON ALL NODES

노드 그룹 작성시 데이터베이스에 정의된 모든 파티션(db2nodes.cfg 파일)에 노드 그룹이 정의되도록 지정합니다.

파티션이 데이터베이스에 추가되면, ALTER NODEGROUP문은 노드그룹(IBMDEFAULTGROUP 포함)에 새 파티션을 포함하도록 실행되어야 합니다. 게다가, 데이터를 파티션으로 옮길 수 있도록 REDISTRIBUTE NODEGROUP 명령을 실행해야 합니다. *Administrative API Reference* 또는 *Command Reference*에서 좀더 정보를 참조하십시오.

ON NODES

노드 그룹에 있는 특정 파티션을 지정합니다. NODE는 NODES와 동의어입니다.

노드 *번호1*

특정 파티션 번호를 지정합니다. [72]

TO 노드 *번호2*

파티션 번호의 범위를 지정합니다. 노드 *번호2*의 값은 노드 *번호1*의 값보다 *크거나* 같아야 합니다(SQLSTATE 428A9). 지정된 파티션 번호 사이의 모든 파티션은 노드 그룹에 포함됩니다.

규칙

- 번호별로 지정되는 파티션이나 노드는 `db2nodes.cfg` 파일에 정의되어야 합니다(SQLSTATE 42729).
- ON NODES절에 나열된 각 노드 *번호*는 한 번 정도 표시됩니다(SQLSTATE 42728).
- 유효한 노드 *번호*는 0-999사이입니다(SQLSTATE 42729).

주

- 이 명령문은 노드 그룹에 대한 파티션 맵을 작성합니다 (더 자세한 사항은 68 페이지의 『여러 파티션에 걸친 데이터 파티션』 참조). 각 파티션 맵에 대해 파티션 맵 식별자(PMAP_ID)가 생성됩니다. 이 정보는 카탈로그에 기록되며, SYSCAT.NODEGROUPS와 SYSCAT.PARTITIONMAPS에서 검색할 수 있습니다. 파티션 맵의 각 항목은 해쉬된 모든 행이 상주하는 목표 파티션을 지정합니다. 단일 파티션 노드 그룹의 경우, 해당하는 파티션 맵에는 항목이 하나 뿐

72. 이전 버전과 호환하도록 'NODEnnnnn' 형태의 노드 이름을 지정할 수 있습니다.

입니다. 다중 파티션 노드 그룹의 경우, 해당하는 파티션 맵에는 4 096 항목이 있으며, 여기서 기본적으로 파티션 번호가 라운드 로빈 방식으로 맵 항목에 할당됩니다.

예

파티션 0, 1, 2, 5, 7, 8이 정의되어 있는 6 파티션 데이터베이스가 있다고 가정합시다.

- 여섯 개의 모든 파티션에서 노드 그룹 호출 MAXGROUP을 작성하고자 한다고 가정합시다. 명령문은 다음과 같습니다.

```
CREATE NODEGROUP MAXGROUP
ON ALL NODES
```

- 파티션 0, 1, 2, 5, 8에서 노드 그룹 MEDGROUP을 작성하고자 한다고 가정합시다. 명령문은 다음과 같습니다.

```
CREATE NODEGROUP MEDGROUP
 ON NODES (0 TO 2, 5, 8)
```

- 파티션(또는 노드) 7에서 단일 파티션 노드 그룹 MINGROUP을 작성하고자 한다고 가정합시다. 명령문은 다음과 같습니다.

```
CREATE NODEGROUP MINGROUP
 ON NODE (7)
```

주: 키워드 NODES의 단수 형태도 승인됩니다.

CREATE PROCEDURE

이 명령문은 응용프로그램 서버(AS)에 저장 프로시듀어를 등록하는 데 사용됩니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권에 다음 중 최소한 하나는 포함되어야 합니다.

- SYSADM 또는 DBADM 권한
- 프로시듀어의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 프로시듀어의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

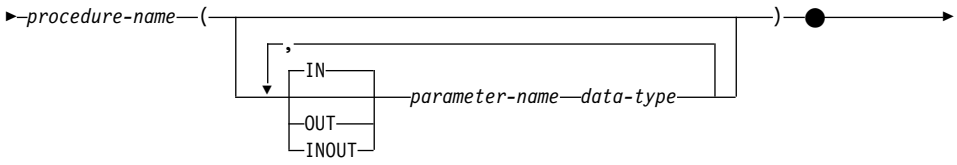
비분리 저장 프로시듀어를 작성하려면, 명령문의 권한 부여 ID가 보유한 특권으로 다음 중 적어도 하나가 포함되어야 합니다.

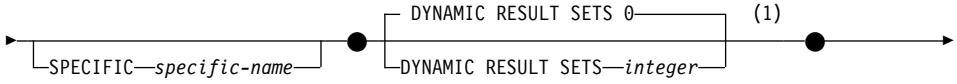
- 데이터베이스에서의 CREATE_NOT_FENCED 권한
- SYSADM 또는 DBADM 권한

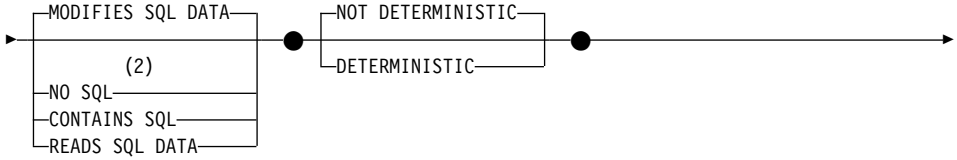
분리 저장 프로시듀어를 작성하는 데, 추가 권한이나 특권이 필요하지는 않습니다.

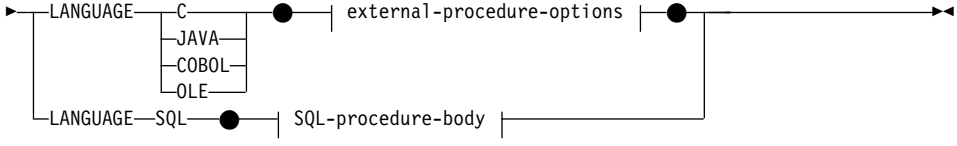
권한 부여 ID가 작업을 수행할 권한이 부족한 경우, 오류(SQLSTATE 42502)가 발생합니다.

구문

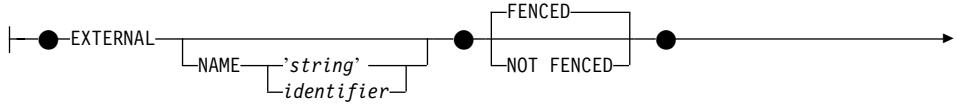


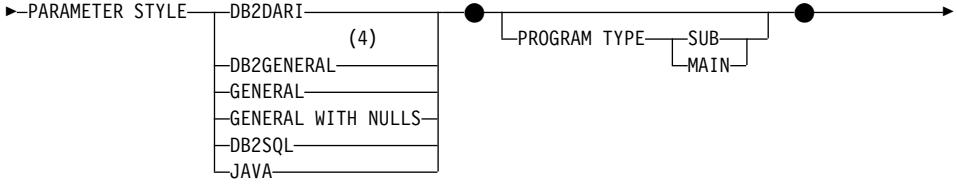


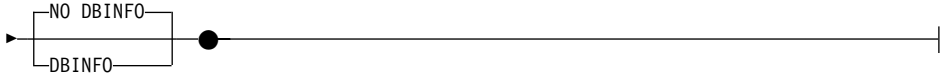




external-procedure-options:







SQL-procedure-body:

주:

1 DYNAMIC RESULT SETS 대신 RESULT SETS를 지정할 수 있습니다.

2 NO SQL은 LANGUAGE SQL에 대해 유효한 선택사항이 아닙니다.

3 CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있습니다.

4 DB2GENERAL 대신 DB2GENRL을 지정할 수 있고 GENERAL 대신 SIMPLE CALL을 지정할 수 있으며 GENERAL WITH NULLS 대신 SIMPLE CALL WITH NULLS를 지정할 수 있습니다.

설명

프로시듀어 이름

정의되고 있는 프로시듀어에 이름을 지정합니다. 이는 프로시듀어를 지정하는 규정화되거나 규정화되지 않은 이름입니다. *프로시듀어 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 128)입니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다.

매개변수의 수와 더불어 암시적 또는 명시적 규정자를 포함하는 이름은 카탈로그에 기술된 프로시듀어를 나타내면 안됩니다(SQLSTATE 42723). 매개변수의 개수가 부가된(스키마 내에서 고유한) 규정화되지 않은 이름은 스키마 전반에 걸쳐 고유하지 않아도 됩니다.

두 부분으로 된 이름이 지정되면, *스키마 이름*은 "SYS"로 시작할 수 없습니다. 그렇지 않으면, 오류(SQLSTATE 42939)가 발생합니다.

(IN | OUT | INOUT 매개변수 이름 데이터 유형,...)

프로시듀어의 매개변수를 식별하고 각 매개변수의 모드, 이름 및 데이터 유형을 지정합니다. 프로시듀어에 예상되는 각 매개변수에 대해 목록에 있는 한 항목이 지정되어야 합니다.

매개변수가 없는 프로시듀어를 등록할 수도 있습니다. 이러한 경우, 데이터 유형이 없는 상태로 괄호를 입력해야 합니다. 예:

```
CREATE PROCEDURE SUBWOOFER() ...
```

한 스키마 내에서 동일하게 명명된 두 프로시듀어는 같은 수의 매개변수를 가질 수 없습니다. 길이, 정밀도 및 스케일은 이 유형의 비교에서 고려되지 않습니다. 그러므로, CHAR(8) 및 CHAR(35)는 같은 유형으로 간주되고, DECIMAL(11,2) 및 DECIMAL (4,3)도 마찬가지입니다. DECIMAL 및 NUMERIC과 같이 이러한 목적을 위해 동일한 유형으로 처리되도록 하는 유형들이 더 있습니다. 시그니처가 중복되면 SQL 오류(SQLSTATE 42723)가 발생합니다.

예를 들어, 다음 명령문이 제공되면,

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART_NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

데이터 유형은 달라도 프로시듀어의 매개변수 수가 동일하므로 두번째 명령문은 실패합니다.

IN | OUT | INOUT

매개변수의 모드를 지정합니다.

- IN - 매개변수가 입력 전용입니다.
- OUT - 매개변수가 출력 전용입니다.
- INOUT - 매개변수가 입력 및 출력용입니다.

매개변수 이름

매개변수의 이름을 지정합니다.

데이터 유형

매개변수의 데이터 유형을 지정합니다.

- CREATE TABLE문의 *데이터 유형* 정의에 지정할 수 있고 프로시듀어를 작성하는 데 사용되는 언어에 해당 항목이 있는 SQL 데이터 유형 스펙과 약어를 지정할 수 있습니다. 저장 프로시듀어와 관련하여 SQL 데이터 유형과 호스트 언어 데이터 유형 사이의 맵핑에 관한 자세한 사항은 *응용프로그램 개발 안내서*의 해당 언어 부분을 참조하십시오.
- 사용자 정의 데이터 유형은 지원되지 않습니다(SQLSTATE 42601).

SPECIFIC *특정 이름*

정의되는 프로시듀어의 인스턴스에 대해 고유한 이름을 제공합니다. 이 특정 이름은 프로시듀어를 삭제할 때 또는 프로시듀어에 주석을 붙일 때 사용할 수 있습니다. 프로시듀어를 호출하는 데에는 사용할 수 없습니다. *특정 이름*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18)입니다. 규정화된 양식은 마침표와 SQL 식별자가 후속되는 *스키마 이름*입니다. 암시적 또는 명시적 규정자를 포함하는 이름은 응용프로그램 서버에 존재하는 다른 프로시듀어 인스턴스를 나타내서는 안 됩니다. 그렇지 않으면 오류(SQLSTATE 42710)가 발생합니다.

*특정 이름*은 기존의 *프로시듀어 이름*과 같을 수 있습니다.

어떤 규정자도 지정하지 않으면, *프로시듀어 이름*에 대해 사용되었던 규정자가 사용됩니다. 규정자가 지정되면, *프로시듀어 이름*의 명시적 또는 암시적 규정자와 같아야 합니다. 그렇지 않으면 오류(SQLSTATE 42882)가 발생합니다.

*특정 이름*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유 이름은 문자 시간소인이 뒤에 오는 SQL 즉, SQLyymmddhhmmsshhn입니다.

DYNAMIC RESULT SETS *정수*

저장 프로시듀어에 대해 리턴된 결과 집합의 산정된 상계(upper bound)를 나타냅니다. 자세한 정보는 *SQL 참조서*에서 "저장 프로시듀어에서 결과 세트 리턴"을 참조하십시오.

RESULT SETS 값이 역방향의 계열 호환성을 위해 DYNAMIC RESULT SETS에 대한 동의어로서 사용될 수도 있습니다.

NO SQL, CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA

저장 프로시듀어가 SQL문과 유형을 발행하는 지를 나타냅니다.

NO SQL

저장 프로시듀어가 SQL문을 초과할 수 없음을 나타냅니다(SQLSTATE 38001).

CONTAINS SQL

SQL 데이터를 읽지도, 수정하지도 않는 SQL문이 저장 프로시듀어에 의해 실행될 수 있음을 나타냅니다(SQLSTATE 38004 또는 42985). 저장 프로시듀어에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

READS SQL DATA

SQL 데이터를 수정하지 않는 SQL문이 저장 프로시듀어에 포함될 수 있음을 나타냅니다(SQLSTATE 38002 또는 42985). 저장 프로시듀어에서 지원되지 않는 명령문은 다른 오류를 리턴합니다(SQLSTATE 38003 또는 42985).

MODIFIES SQL DATA

저장 프로시듀어에서 지원되지 않는 명령문을 제외하고, 저장 프로시듀어가 SQL문을 실행할 수 있음을 나타냅니다(SQLSTATE 38003 또는 42985).

다음 표는 지정된 SQL 데이터 액세스 표시가 있는 저장 프로시듀어에서 SQL문(첫번째 컬럼에 지정된)을 실행할 수 있는 지를 나타냅니다. 실행 가능한 SQL문이 NO SQL로 정의된 저장 프로시듀어에서 발견되면, SQLSTATE 38001이 리턴됩니다. 다른 실행 문맥의 경우, 문맥에서 지원되지 않는 SQL문은 SQLSTATE 38003을 리턴합니다. CONTAINS SQL 문맥에서 허용되지 않는 다른 SQL문의 경우, SQLSTATE 38004가 리턴되고 READS SQL DATA 문맥에서는 SQLSTATE 38002가 리턴됩니다. SQL 프로시듀어를 작성하는 동안, SQL 데이터 액세스 표시와 일치하지 않는 명령문은 SQLSTATE 42895가 리턴되도록 합니다.

표 21. SQL문 및 SQL 데이터 액세스 표시

SQL문	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
ALTER...	N	N	N	Y
BEGIN DECLARE SECTION	Y(1)	Y	Y	Y
CALL	N	Y(4)	Y(4)	Y(4)
CLOSE CURSOR	N	N	Y	Y
COMMENT ON	N	N	N	Y
COMMIT	N	N	N	N
COMPOUND SQL	N	Y	Y	Y
CONNECT(2)	N	N	N	N
CREATE	N	N	N	Y
DECLARE CURSOR	Y(1)	Y	Y	Y
DECLARE GLOBAL TEMPORARY TABLE	N	Y	Y	Y
DELETE	N	N	N	Y
DESCRIBE	N	N	Y	Y
DISCONNECT(2)	N	N	N	N
DROP ...	N	N	N	Y
END DECLARE SECTION	Y(1)	Y	Y	Y
EXECUTE	N	Y(3)	Y(3)	Y
EXECUTE IMMEDIATE	N	Y(3)	Y(3)	Y
EXPLAIN	N	N	N	Y
FETCH	N	N	Y	Y
FREE LOCATOR	N	Y	Y	Y
FLUSH EVENT MONITOR	N	N	N	Y
GRANT ...	N	N	N	Y
INCLUDE	Y(1)	Y	Y	Y
INSERT	N	N	N	Y
LOCK TABLE	N	Y	Y	Y
OPEN CURSOR	N	N	Y	Y
PREPARE	N	Y	Y	Y
REFRESH TABLE	N	N	N	Y
RELEASE CONNECTION(2)	N	N	N	N

표 21. SQL문 및 SQL 데이터 액세스 표시 (계속)

SQL문	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
RELEASE SAVEPOINT	N	N	N	Y
RENAME TABLE	N	N	N	Y
REVOKE ...	N	N	N	Y
ROLLBACK	N	Y	Y	Y
ROLLBACK TO SAVEPOINT	N	N	N	Y
SAVEPOINT	N	N	N	Y
SELECT INTO	N	N	Y	Y
SET CONNECTION(2)	N	N	N	N
SET INTEGRITY	N	N	N	Y
SET 특수 레지스터	N	Y	Y	Y
UPDATE	N	N	N	Y
VALUES INTO	N	N	Y	Y
WHENEVER	Y(1)	Y	Y	Y

주:

1. NO SQL 옵션에는 어떤 SQL문도 지정할 수 없다는 것이 내포되어 있지만, 실행할 수 없는 명령문은 제한되지 않습니다.
2. 연결 관리 명령문은 저장 프로시듀어 실행 문맥에서 허용되지 않습니다.
3. 이것은 실행되는 명령문에 따라 다릅니다. EXECUTE문에 대해 지정된 명령문은 적용되는 특정 SQL 액세스 레벨의 문맥에서 허용되는 명령문이어야 합니다. 예를 들어, 적용되는 SQL 액세스 레벨이 READS SQL DATA일 경우, 명령문은 INSERT, UPDATE 또는 DELETE가 아니어야 합니다.
4. 저장 프로시듀어의 CALL문은 호출하는 저장 프로시듀어와 같은 프로그래밍 언어로 작성된 저장 프로시듀어만 참조할 수 있습니다.

LANGUAGE

이 필수 절은 저장 프로시듀어의 본체가 작성되는 언어 인터페이스 규약을 지정하는 데 사용됩니다.

C 데이터베이스 관리 프로그램은 C 프로시듀어인 것처럼 저장 프로시듀

어를 호출합니다. 저장 프로시듀어는 표준 ANSI C 프로토타입에 의해 정의된 대로 C 언어 호출 규칙과 연계 규칙에 따라야 합니다.

JAVA 이것은 데이터베이스 관리 프로그램이 Java 클래스의 한 방법으로 저장 프로시듀어를 호출하는 것을 의미합니다.

COBOL

데이터베이스 관리 프로그램은 COBOL 프로시듀어인 것처럼 저장 프로시듀어를 호출합니다.

OLE 데이터베이스 관리 프로그램은 OLE 자동화 오브젝트에 의해 노출된 방법인 것처럼 저장 프로시듀어를 호출합니다. 저장 프로시듀어는 OLE 자동화 데이터 및 호출 메카니즘과 일치해야 합니다. 또한, OLE 자동화 오브젝트는 처리 중인 서버로 구현되어야 합니다(DLL). 이러한 제한사항은 OLE Automation Programmer's Reference에 요약되어 있습니다.

LANGUAGE OLE는 Windows 32 비트 운영 체제용 DB2에 저장된 저장 프로시듀어에 대해서만 지원됩니다.

SQL 지정된 *SQL 프로시듀어 본문*에는 저장 프로시듀어 처리를 정의하는 명령문이 포함됩니다.

EXTERNAL

이 절은 CREATE PROCEDURE문이 외부 프로그래밍 언어로 작성된 코드를 기초로 문서화된 연계 규칙과 인터페이스에 따라 새 함수를 등록하는 데 사용됨을 나타냅니다.

NAME절이 지정되지 않을 경우, "NAME *프로시듀어 이름*"이 사용됩니다.

NAME *'문자열'*

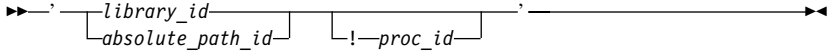
이 절은 정의되는 프로시듀어를 구현하는 사용자 작성 코드의 이름을 식별합니다.

'문자열' 옵션은 최대 254 문자로 된 문자열 상수입니다. 문자열에 사용되는 형식은 지정되는 LANGUAGE에 따라 다릅니다.

- LANGUAGE C의 경우:

  지정된 *문자열*은 라이브러리 이름과 그 라이브러리내의 프로시듀어로써, 작성중인 저장 프로시듀어를 실행하기 위해 데이터베이스 관리 프로그

램이 호출합니다. CREATE PROCEDURE문이 수행될 때 라이브러리(및 그 라이브러리내의 프로시듀어)가 존재하지 않아도 됩니다. 그러나, 프로시듀어가 호출될 때 라이브러리와 그 라이브러리 내의 프로시듀어가 존재해야 하며 데이터베이스 서버 머신에 액세스할 수 있어야 합니다.



이름은 작은 따옴표로 묶어야 합니다. 작은 따옴표 내에서 공백은 허용되지 않습니다.

library_id

프로시듀어를 포함하는 라이브러리 이름을 식별합니다. 데이터베이스 관리 프로그램은 .../sqllib/function/비분리 디렉토리 및 .../sqllib/함수 디렉토리(UNIX용 시스템), 또는 ...*instance_name*\function\비분리 디렉토리 및 ...*instance_name*\함수 디렉토리(OS/2, 그리고 DB2INSTPROF 레지스트리 변수에 지정된 Windows 32 비트 운영 체제)에서 그 라이브러리를 찾습니다. 이곳은 데이터베이스 관리 프로그램을 수행하기 위해 사용되는 제어하는 sqllib 디렉토리를 데이터베이스 관리 프로그램이 찾는 곳입니다. 예를 들어, UNIX 기본 시스템의 제어 sqllib 디렉토리는 /u/$DB2INSTANCE/sqllib입니다.

'myproc'이 UNIX 기본 시스템의 *library_id*이면, 이로 인해 데이터베이스 관리 프로그램이 /u/production에서 수행될 경우에 라이브러리 /u/production/sqllib/function/unfenced/myfunc 및 /u/production/sqllib/function/myfunc에서 함수를 찾게 됩니다.

OS/2, Windows 32 비트 운영 체제의 경우, 데이터베이스 관리 프로그램은 *library_id*가 함수 디렉토리에 위치되지 않은 경우에 분리된 것을 수행됩니다.

이 디렉토리의 아무 곳에나 위치되어 있는 저장 프로시듀어는 등록 속성을 사용하지 못합니다.

absolute_path_id

프로시듀어의 전체 경로 이름을 식별합니다.

예를 들어, UNIX형 시스템에서 '/u/jchui/mylib/myproc'는 데이터베이스 관리 프로그램이 /u/jchui/mylib에서 myproc 프로시듀어를 찾도록 합니다.

OS/2에서, Windows 32 비트 운영 체제 'd:\mylib\myproc'는 데이터베이스 관리 프로그램이 d:\mylib 디렉토리로부터 myproc.dll 파일을 로드하게 합니다.

절대 경로를 지정할 경우, 프로시듀어는 FENCED나 NOT FENCED 속성을 무시하고 분리된 것으로 수행합니다.

! proc_id

호출될 프로시듀어의 진입점 이름을 식별합니다. !는 library id와 함수 id 사이의 분리 문자로 사용됩니다. *! proc_id*가 생략되면, 데이터베이스 관리 프로그램은 라이브러리가 링크될 때 설정된 기본 진입점을 사용합니다.

예를 들어, UNIX용 시스템에서 'mymod!proc8'는 데이터베이스 관리 프로그램이 라이브러리 $inst_home_dir/sqllib/function/mymod를 찾아 그 라이브러리 내에서 진입점 proc8을 사용하도록 지시합니다.

OS/2에서, Windows 32 비트 운영 체제 'mymod!proc8'은 데이터베이스 관리 프로그램이 mymod.dll 파일을 로드하고 동적 링크 라이브러리(DLL)에서 proc8() 프로시듀어를 호출하게 합니다.

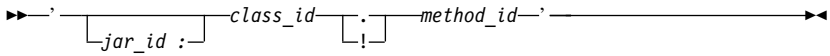
문자열이 적절히 형성되지 않으면, 오류(SQLSTATE 42878)가 발생합니다.

모든 저장 프로시듀어의 본체는 데이터베이스의 각 파티션에 마운트되어 사용 가능한 디렉토리에 있어야 합니다.

- LANGUAGE JAVA의 경우:

  지정된 *문자열*에는 선택적 jar 파일 식별자, 클래스 식별자 및 메소드 식별자가 포함됩니다. 이는 작성되는 저장 프로시듀어를 실행하기 위해 데이터베이스 관리 프로그램이 호출하는 것입니다. CREATE PROCEDURE문이 수행될 때 클래스 식별자와 방법 식별자는 존재하지 않아도 됩니다. *jar_id*를 지정할 경우, CREATE PROCEDURE문이 수행될 때 이것이 존재해야 합니다. 그러나, 프로시듀어가 호출될 경

우, 클래스 식별자와 메소드 식별자가 있어야 하며 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다. 그렇지 않으면, 오류가 발생합니다(SQLSTATE 42884).



이름은 작은 따옴표로 묶어야 합니다. 작은 따옴표 내에서 공백은 허용되지 않습니다.

jar_id

데이터베이스에 설치될 때 jar 콜렉션에 부여된 jar 식별자를 식별합니다. 간단한 식별자 또는 스키마 규정화된 식별자일 수 있습니다. 'myJar' 및 'mySchema.myJar'가 그 예입니다.

class_id

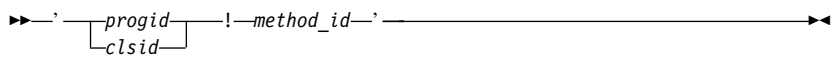
Java 오브젝트의 클래스 식별자를 나타냅니다. 클래스가 패키지의 일부인 경우, 클래스 식별자 부분에 완전한 패키지 접두어, 예를 들어, 'myPacks.StoredProcs'가 포함되어야 합니다. Java 가상 머신은 클래스를 디렉토리 '../myPacks/StoredProcs/'에서 보게 됩니다. OS/2, Windows 32 비트 운영 체제의 경우, Java 가상 머신은 디렉토리 '..\myPacks\StoredProcs\'에서 보게 됩니다.

method_id

호출할 Java 클래스의 방법 이름을 나타냅니다.

- LANGUAGE OLE의 경우:

  지정된 문자열은 OLE 프로그램가능 식별자(*progid*) 또는 클래스 식별자(*clsid*)와 방법 식별자(*method_id*)로서, 명령문에 의해 작성중인 저장 프로시듀어를 실행하기 위해 데이터베이스 관리 프로그램이 호출한 것입니다. CREATE PROCEDURE문이 실행될 때 프로그램가능 식별자, 클래스 식별자 및 방법 식별자는 존재하지 않아도 됩니다. 그러나, 프로시듀어가 CALL문에서 사용될 경우, 메소드 식별자가 있어야 하며 데이터베이스 서버 머신에서 액세스할 수 있어야 합니다. 그렇지 않으면, 오류가 발생합니다(SQLSTATE 42724).



이름은 작은 따옴표로 묶어야 합니다. 작은 따옴표 내에서 공백은 허용되지 않습니다.

progid

OLE 오브젝트의 프로그램가능 식별자를 나타냅니다.

*progid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임 OLE 자동 제어기로 이송됩니다. 지정된 OLE 오브젝트는 작성 가능해야 하며 후기 바인딩(IDispatch형 바인딩이라고도 함)을 지원해야 합니다. 규정에 의해, progids의 형식은 다음과 같습니다.

<program_name>.<component_name>.<version>

이것이 유일한 규정으로 반드시 지켜야 할 규칙은 아니므로, 사실상 *progids*은 다른 형식을 취할 수도 있습니다.

clsid

작성할 OLE 오브젝트의 클래스 식별자를 나타냅니다. OLE 오브젝트가 progid를 사용하여 등록되지 않은 경우 progid를 지정하기 위한 대안으로 사용할 수 있습니다. *clsid*은 다음과 같은 형식을 갖습니다.

{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}

여기서, 'n'은 영숫자 문자입니다. *clsid*는 데이터베이스 관리 프로그램에 의해 해석되지 않으며, 런타임 OLE API로 이송됩니다.

method_id

호출할 OLE 오브젝트의 방법 이름을 나타냅니다.

NAME *식별자*

지정된 이 *식별자*는 SQL 식별자입니다. SQL 식별자는 문자열에서 *library id*로 사용됩니다. 분리 식별자가 아닐 경우, 식별자는 대문자로 겹쳐집니다. 식별자에 스키마 이름이 규정된 경우, 스키마 이름 부분은 무시됩니다. 이 형식의 이름은 LANGUAGE C에서만 사용할 수 있습니다.

FENCED 또는 NOT FENCED

이 절은 저장 프로시듀어가 데이터베이스 관리 프로그램 운영 환경의 프로세스나 주소 공간을 수행하기에 『안전한』지(NOT FENCED) 않은지(FENCED)를 지정합니다.

저장 프로시듀어가 FENCED로 등록되면, 데이터베이스 관리 프로그램은 프로시듀어에서 액세스를 못하도록 내부 자원(예를 들어, 데이터 버퍼)을 분리시킵니다. 모든 프로시듀어는 FENCED나 NOT FENCED로 수행되는 옵션을 가지고 있습니다. 일반적으로, FENCED로 수행되는 프로시듀어는 NOT FENCED로 수행되는 프로시듀어와 다르게 수행됩니다.

저장 프로시듀어가 .../sqllib/function/unfenced 디렉토리와 .../sqllib/function 디렉토리(UNIX 기반 시스템) 또는 ...*instance_name*\function\unfenced 디렉토리와 ...*instance_name*\function 디렉토리(OS/2, Windows 32 비트 운영 체제)에 위치되면 이 때 등록된 FENCED 또는 NOT FENCED 속성(및 다른 모든 등록된 속성)은 무시됩니다.

주: 적절히 점검되지 않는 프로시듀어에 대해 NOT FENCED를 사용하면 DB2의 무결성을 보완할 수 있습니다. DB2는 발생할 수도 있는 많은 일반적 장애에 대한 예방책을 취하나, NOT FENCED 저장 프로시듀어가 사용될 때 완전한 무결성을 보증할 수는 없습니다.

FENCED에서 NOT FENCED로 변경하려면, (먼저 그 프로시듀어를 삭제한 후 다시 작성하여) 프로시듀어를 다시 등록해야 합니다. 저장 프로시듀어를 NOT FENCED로 등록하려면 SYSADM 권한, DBADM 권한 또는 특별 권한(CREATE_NOT_FENCED)이 필요합니다. LANGUAGE OLE에서는 함수에 대해 FENCED만을 지정할 수 있습니다.

PARAMETER STYLE

이 절은 저장 프로시듀어로부터 값을 리턴하고 매개변수를 전달하기 위한 규약을 지정하는 데 사용됩니다.

DB2DARI

이것은 저장 프로시듀어가 C언어 호출 및 연결에 적합하며 매개변수를 전달하는 데 사용된 규약을 사용함을 의미합니다. 이것은 LANGUAGE C를 사용할 때만 지정할 수 있습니다.

DB2GENERAL

이것은 저장 프로시듀어가 Java 방법을 사용하여 정의된, 매개변수를 전달하는 데 사용된 규약을 사용함을 의미합니다. 이것은 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

값 DB2GENRL는 DB2GENERAL의 동의어로 사용할 수 있습니다.

GENERAL

이는 저장 프로시듀어가 CALL에서 지정된 매개변수를 수신하는 매개변수 전달 메카니즘을 사용할 것임을 의미합니다. 매개변수는 언어가 기대하는 대로 직접 전달되며, SQLDA 구조는 사용되지 않습니다. 이것은 LANGUAGE C 또는 COBOL을 사용할 때만 지정할 수 있습니다.

널(NULL) 표시기는 프로그램에 직접 전달되지 않습니다.

값 SIMPLE CALL은 GENERAL의 동의어로 사용할 수 있습니다.

GENERAL WITH NULLS

GENERAL에서 지정된 대로 CALL문상의 매개변수외에, 또 다른 인수가 저장 프로시듀어에 전달됩니다. 이 추가 인수는 CALL문상의 각 매개변수에 대한 널(NULL) 표시기의 벡터를 포함합니다. C에서는, 이것이 short ints의 배열일 것입니다. 이것은 LANGUAGE C 또는 COBOL을 사용할 때만 지정할 수 있습니다.

값 SIMPLE CALL WITH NULLS는 GENERAL WITH NULLLS의 동의어로 사용할 수 있습니다.

DB2SQL

CALL문의 매개변수 외에도, 다음 인수들이 저장 프로시듀어에 전달됩니다.

- CALL문상의 각 매개변수에 대한 널(NULL) 표시기
- DB2에 리턴될 SQLSTATE
- 저장 프로시듀어의 규정화된 이름

- 저장 프로시듀어의 특정 이름
- DB2에 리턴될 SQL 진단 문자열

이것은 LANGUAGE C, COBOL 또는 OLE를 사용할 때만 지정할 수 있습니다.

JAVA 이는 저장 프로시듀어가 Java 언어 및 SQLJ 루틴 스펙을 준수하는 매개변수 전달 규약을 사용할 것임을 의미합니다. 값 리턴을 쉽게 하기 위해 IN/OUT 및 OUT 매개변수가 단일 항목 배열로서 전달될 것입니다. 이것은 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

PARAMETER STYLE JAVA 프로시듀어는 DBINFO 또는 PROGRAM TYPE절을 지원하지 않습니다.

매개변수를 전달하는 방법에 대한 *응용프로그램 개발 안내서*에서 자세한 내용을 참조하십시오.

PROGRAM TYPE

저장된 절차는 주요 루틴이나 서브 루틴의 스타일로 매개변수를 기대하는지의 여부를 지정합니다.

SUB

저장된 절차는 서로 다른 인수로 전달될 매개변수를 기대합니다.

MAIN

저장된 절차는 인수 카운터 그리고 인수 요소(argc, argv)로서 전달될 매개변수를 기대합니다. 호출될 저장된 절차의 이름은 ″main″이 되어야 합니다. 이러한 유형의 저장 프로시듀어는 독립형 실행 파일과는 반대로 공유 라이브러리와 같은 형태로 계속 내장되어야 합니다.

PROGRAM TYPE에 대한 기본값은 SUB입니다. PROGRAM TYPE MAIN은 LANGUAGE C 또는 COBOL과 PARAMETER STYLE GENERAL, GENERAL WITH NULLS 또는 DB2SQL에 대해서만 유효합니다.

DETERMINISTIC 또는 **NOT DETERMINISTIC**

이 절은 프로시듀어가 항상 주어진 값에 대해 동일한 결과를 리턴하는지

(DETERMINISTIC) 또는 프로시듀어가 결과에 영향을 미치는 동일한 상태 값에 의존하는지(NOT DETERMINISTIC) 여부를 지정합니다. 즉, DETERMINISTIC 프로시듀어는 입력이 동일한 연속된 호출에 대해 항상 동일한 결과를 리턴해야 합니다.

이 절은 현재 저장 프로시듀어의 처리에는 영향을 주지 않습니다.

CALLED ON NULL INPUT

CALLED ON NULL INPUT는 항상 저장 프로시듀어에 적용됩니다. 이것은 어떠한 인수가 널(NULL) 값이더라도 상관이 없음을 의미합니다. 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 널(NULL) 인수 값에 대한 테스트 책임은 저장된 절차에 있습니다.

NULL CALL이 역방향의 계열 호환성을 위해 CALLED ON NULL INPUT에 대한 동의어로서 사용될 수도 있습니다.

NO DBINFO 또는 DBINFO

DB2가 확인한 특정 정보가 추가 호출 인수로 호출되는지(DBINFO) 않은지(NO DBINFO)의 여부를 지정합니다. NO DBINFO가 기본값입니다. DBINFO는 LANGUAGE OLE에서는 지원하지 않습니다(SQLSTATE 42613). 이것은 또한 PARAMETER STYLE JAVA, DB2GENERAL 또는 DB2DARI에 대해서도 지원되지 않습니다.

DBINFO가 지정되는 경우, 다음 정보가 들어 있는 저장 프로시듀어로 구조가 전달됩니다.

- 데이터베이스 이름 - 현재 연결된 데이터베이스의 이름.
- 응용프로그램 ID - 데이터베이스로의 각 연결에 형성되는 고유한 응용프로그램 ID.
- 응용프로그램 권한 부여 ID - 응용프로그램 런타임 권한 부여 ID.
- 코드 페이지 - 데이터베이스 코드 페이지를 식별합니다.
- 스키마 이름 - 저장된 절차에는 적용되지 않음
- 테이블 이름 - 저장된 절차에는 적용되지 않습니다.
- 컬럼 이름 - 저장된 절차에는 적용되지 않습니다.

- 데이터베이스 버전/릴리스 - 저장된 절차를 호출하는 데이터베이스 서버의 버전, 릴리스 및 개정 레벨을 나타냅니다.
- 플랫폼 - 서버의 플랫폼 유형을 포함합니다.
- 테이블 함수 결과 컬럼 수 - 저장된 절차에는 적용되지않음.

저정된 절차로 전달되는 방법과 구조에 대한 *응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.

SQL 프로시듀어 내용

SQL 프로시듀어의 내용인 SQL 명령문을 지정합니다. 여러 SQL 프로시듀어 명령문을 복합 명령문 내에 지정할 수 있습니다. 1207 페이지의 『제7장 SQL 프로시듀어』에서 자세한 설명을 참조하십시오.

주

- 저장 프로시듀어의 프로그램 작성에 대하여는 *응용프로그램 개발 안내서*에서 참조하십시오.
- 아직 존재하지 않는 스키마 이름을 사용하여 색인을 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 호출자의 특수 레지스터에 대한 설정은 호출 시 저장 프로시듀어에 의해 계승되고 호출자에게 리턴 시 복원됩니다. 특수 레지스터는 저장 프로시듀어 내에서 변경될 수 있지만, 이러한 변경사항이 호출자에게 영향을 주지는 않습니다. 이는 프로시듀어의 특수 레지스터에 대해 수행된 변경사항이 호출자에 대한 설정이 되는 레거시 저장 프로시듀어(매개변수 스타일 DB2DARI로 정의되거나 기본 라이브러리에서 저장되는 저장 프로시듀어)의 경우에는 적용되지 않습니다.

예

예 1: 현재 사용 가능한 양과 부품의 비용을 리턴하며 부품 번호를 통과하는, 자바에 기록된 저장 프로시듀어에 대해 프로시듀어 정의를 작성합니다.

```
CREATE PROCEDURE PARTS_ON_HAND (IN  PARTNUM  INTEGER,
                                OUT COST     DECIMAL(7,2),
                                OUT QUANTITY INTEGER)
                EXTERNAL NAME 'parts.onhand'
                LANGUAGE JAVA PARAMETER STYLE JAVA
```

예 2: 어셈블리 번호를 통과하며 어셈블리를 구성하는 부품 수, 부품의 총 비용 그리고 부품 번호, 수량, 각 부품의 단가를 나열하는 결과 집합을 되돌리는 C에 기록된 저장 프로시듀어에 대해 프로시듀어 정의를 작성합니다.

```
CREATE PROCEDURE ASSEMBLY_PARTS  (IN  ASSEMBLY_NUM  INTEGER,
                                  OUT NUM_PARTS     INTEGER,
                                  OUT COST  DOUBLE)
       EXTERNAL NAME 'parts!assembly'
       DYNAMIC RESULT SETS 1 NOT FENCED
       LANGUAGE C PARAMETER STYLE GENERAL
```

예 3: 중간 직원 급여를 리턴하는 SQL 프로시듀어를 작성합니다. 중간 급여보다 많이 받는 모든 사원들의 이름, 지위 및 급여를 포함하는 결과 세트를 리턴합니다.

```
CREATE PROCEDURE MEDIAN_RESULT_SET
(OUT medianSalary DOUBLE)
     RESULT SETS 1
     LANGUAGE SQL
BEGIN
     DECLARE v_numRecords INT DEFAULT 1;
     DECLARE v_counter INT DEFAULT 0;
     DECLARE c1 CURSOR FOR
        SELECT CAST(salary AS DOUBLE)
           FROM staff
           ORDER BY salary;
     DECLARE c2 CURSOR WITH RETURN FOR
        SELECT name, job, CAST(salary  AS INTEGER)
           FROM staff
           WHERE salary > medianSalary
           ORDER BY salary;
     DECLARE EXIT HANDLER FOR NOT FOUND
        SET medianSalary = 6666;
     SET medianSalary = 0;
     SELECT COUNT(*) INTO v_numRecords
        FROM STAFF;
     OPEN c1;
     WHILE v_counter < (v_numRecords / 2 + 1)
        DO FETCH c1 INTO medianSalary;
        SET v_counter = v_counter + 1;
```

```
        END WHILE;
        CLOSE c1;
        OPEN c2;
    END
```

CREATE SCHEMA

CREATE SCHEMA문은 스키마를 정의합니다. 또한 오브젝트를 작성하고 명령문 내에서 오브젝트에 관한 특권을 권한 부여할 수도 있습니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

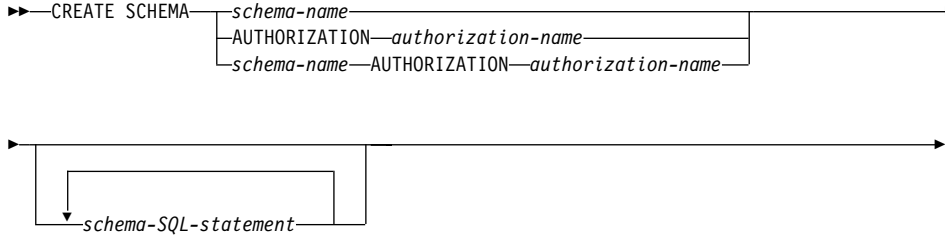
SYSADM 또는 DBADM 권한을 보유한 권한 부여 ID는 유효한 *스키마 이름* 또는 *권한 부여 이름*을 사용하여 스키마를 작성할 수 있습니다.

SYSADM 또는 DBADM 권한을 갖지 않은 권한 부여 ID는 명령문의 권한 부여 ID에 부합하는 *스키마 이름* 또는 *권한 부여 이름*을 사용하여 스키마를 작성할 수 있습니다.

명령문에 *스키마 SQL문*이 들어 있는 경우, *권한 부여 이름*(지정되지 않은 경우, 명령문의 권한 부여 ID로 기본 설정됨)이 보유한 특권에는 다음 중 적어도 하나가 포함되어야 합니다.

- 각각의 *스키마 SQL문*을 수행하는 데 필요한 특권
- SYSADM 또는 DBADM 권한

구문



설명

스키마 이름

*스키마*의 이름을 지정합니다. 이 이름은 카탈로그에 이미 기술되어 있는 *스키마*를 나타내면 안됩니다(SQLSTATE 42710). 이름은 "SYS"로 시작할 수 없습니다(SQLSTATE 42939). *스키마*의 소유자는 명령문을 실행한 권한 부여 ID입니다.

AUTHORIZATION *권한 부여 이름*

*스키마*의 소유자인 사용자를 식별합니다. *권한 부여 이름*은 *스키마*에 이름을 지정하는 데에도 사용됩니다. *권한 부여 이름*은 카탈로그에 이미 기술되어 있는 *스키마*를 나타내면 안 됩니다(SQLSTATE 42710).

스키마 이름 **AUTHORIZATION** *권한 부여 이름*

*권한 부여 이름*의 사용자를 *스키마* 소유자로 하여 *스키마 스키마 이름*을 식별합니다. *스키마 이름*은 카탈로그에 이미 기술되어 있는 *스키마*의 *스키마* 이름을 나타내면 안 됩니다(SQLSTATE 42710). *스키마 이름*은 "SYS"로 시작할 수 없습니다(SQLSTATE 42939).

스키마 SQL문

CREATE SCHEMA문의 일부로 포함시킬 수 있는 SQL문으로는 다음이 있습니다.

- 입력된 테이블 및 요약 테이블은 제외한 CREATE TABLE문(800 페이지의 『CREATE TABLE』 참조)
- 유형화된 뷰를 제외하는 CREATE VIEW문(931 페이지의 『CREATE VIEW』 참조)
- CREATE INDEX문(740 페이지의 『CREATE INDEX』 참조)
- COMMENT ON문(588 페이지의 『COMMENT ON』 참조)
- GRANT문(1054 페이지의 『GRANT(테이블, 뷰 또는 별명 특권)』 참조).

주

- *스키마*의 소유자는 다음과 같이 결정됩니다.
  - AUTHORIZATION절이 지정된 경우, 지정된 *권한 부여 이름*이 *스키마* 소유자입니다.

  - AUTHORIZATION절이 지정되지 않은 경우, CREATE SCHEMA문을 실행한 권한 부여 ID가 스키마 소유자입니다.
- 스키마 소유자는 사용자로 간주됩니다(그룹이 아님).
- CREATE SCHEMA문을 사용하여 스키마가 명시적으로 작성된 경우, 스키마 소유자에게는 스키마에 대해 CREATEIN, DROPIN 및 ALTERIN 특권이 권한 부여되며, 이들 특권을 다른 사용자에게 부여할 권한 부여도 주어집니다.
- CREATE SCHEMA문의 일부로 작성된 오브젝트의 정의자가 스키마 소유자입니다. 스키마 소유자는 CREATE SCHEMA문의 일부인 특권의 권한을 준 사용자이기도 합니다.
- CREATE SCHEMA문의 SQL문에 있는 규정화되지 않은 오브젝트 이름은 작성된 스키마 이름에 의해 내재적으로 규정화됩니다.
- CREATE문에 작성 중인 오브젝트에 대한 규정화 이름이 포함된 경우, 규정화 이름에 지정된 스키마 이름은 작성 중인 스키마의 이름과 같아야 합니다 (SQLSTATE 42875). 명령문내에서 참조된 다른 오브젝트는 다른 유효한 스키마 이름으로 규정화할 수 있습니다.
- AUTHORIZATION절이 지정되고 DCE 인증이 사용되면, 지정된 *권한 부여 이름*은 다음과 같은 절의 명령문 수행에 필요한 권한 부여를 평가하는 데 고려되지 않습니다. 지정된 *권한 부여 이름*이 스키마를 작성한 권한 부여 ID와 다르면, CREATE SCHEMA문 실행중에 권한 부여 실패가 발생합니다.
- 스키마 이름으로 "SESSION"은 사용하지 않도록 하십시오. 선언된 임시 테이블은 "SESSION"에 의해 규정되어야 하므로, 영구 테이블의 이름과 같은 이름의 임시 테이블을 응용프로그램에서 선언되도록 할 수 있습니다. 스키마 이름이 "SESSION"인 테이블을 참조하는 SQL문은 같은 이름의 영구적 테이블이 아니라 선언된 임시 테이블로 분석됩니다(명령문이 컴파일될 때). SQL문은 정적 Embedded SQL과 동적 Embedded SQL문에 대해 서로 다른 시간에 컴파일되므로, 결과는 선언된 임시 테이블이 정의되는 시기에 따라 다릅니다. 지속적 테이블, 뷰 또는 별명이 "SESSION"이라는 스키마 이름으로 정의되지 않은 경우, 다음 사항은 고려하지 않아도 됩니다.

예

예 1: DBADM 권한을 가진 사용자로서, 사용자 RICK을 소유자로 하여 RICK이라는 스키마를 작성합니다.

```
CREATE SCHEMA RICK AUTHORIZATION RICK
```

예 2: 재고 부품표와 부품 번호에 대한 색인을 갖는 스키마를 작성합니다. 테이블에 대한 권한을 JONES에게 부여합니다.

```
CREATE SCHEMA INVENTRY
   CREATE TABLE PART (PARTNO   SMALLINT NOT NULL,
                      DESCR    VARCHAR(24),
                      QUANTITY INTEGER)
   CREATE INDEX PARTIND ON PART (PARTNO)
   GRANT ALL ON PART TO JONES
```

예 3: 다른 테이블을 참조하는 각자의 외부 키를 갖는 두 개의 테이블을 사용하여 PERS라는 스키마를 작성합니다. 이것은 ALTER TABLE문을 사용하지 않고 테이블을 작성할 수 있도록 하는 CREATE SCHEMA문의 특성을 보여주는 예입니다.

```
CREATE SCHEMA PERS
   CREATE TABLE ORG (DEPTNUMB  SMALLINT NOT NULL,
                     DEPTNAME VARCHAR(14),
                     MANAGER  SMALLINT,
                     DIVISION VARCHAR(10),
                     LOCATION VARCHAR(13),
                     CONSTRAINT PKEYDNO
                        PRIMARY KEY (DEPTNUMB),
                     CONSTRAINT FKEYMGR
                        FOREIGN KEY (MANAGER)
                        REFERENCES STAFF (ID) )
   CREATE TABLE STAFF (ID        SMALLINT NOT NULL,
                       NAME      VARCHAR(9),
                       DEPT      SMALLINT,
                       JOB       VARCHAR(5),
                       YEARS     SMALLINT,
                       SALARY    DECIMAL(7,2),
                       COMM      DECIMAL(7,2),
                       CONSTRAINT PKEYID
                          PRIMARY KEY (ID),
                       CONSTRAINT FKEYDNO
                          FOREIGN KEY (DEPT)
                          REFERENCES ORG (DEPTNUMB) )
```

CREATE SERVER

CREATE SERVER문은[73] 데이터 소스를 연합 데이터베이스에 정의합니다.

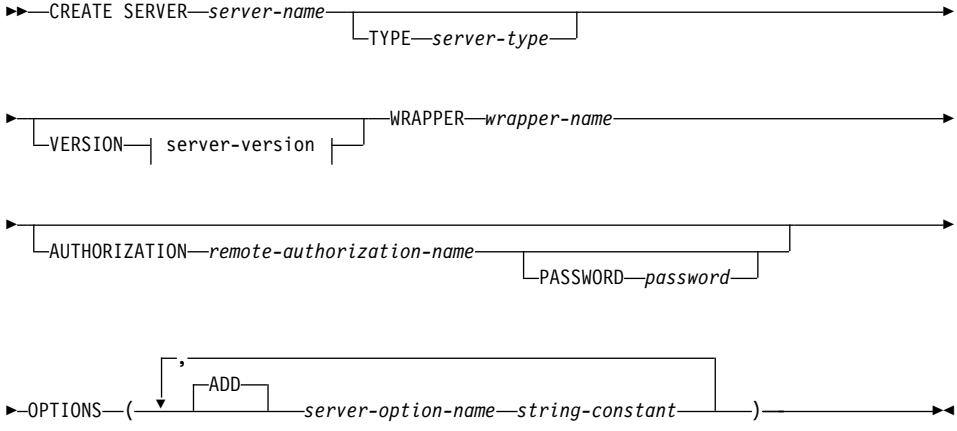
호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

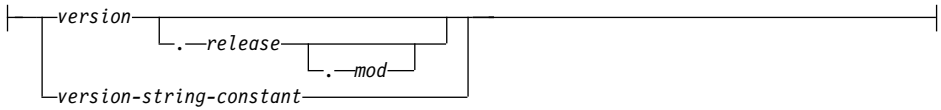
명령문의 권한 부여 ID는 연합 데이터베이스에서 SYSADM 또는 DBADM 권한을 가져야 합니다.

구문



server-version:

73. 이 명령문에서 SERVER라는 용어와 *server*-로 시작하는 매개변수 이름은 연합 시스템에 있는 데이터 소스만 지칭합니다. 그러한 시스템에 있는 연합 서버나 DRDA 응용프로그램 서버(AS)는 지칭하지 않습니다. 연합 시스템에 관한 정보는 47 페이지의 『DB2 연합 시스템』에서 참조하십시오. DRDA 응용프로그램 서버(AS)에 대한 정보는 34 페이지의 『분산 관계형 데이터베이스』에서 참조하십시오.



설명

server-name

연합 데이터베이스에 정의되고 있는 데이터 소스에 이름을 부여합니다. 이름은 카탈로그에 기술되어 있는 데이터 소스를 식별하면 안 됩니다. *server-name*은 연합 데이터베이스에 있는 모든 테이블 공간 이름과 동일해서는 안 됩니다.

TYPE *server-type*

*server-name*으로 표시된 데이터 소스의 유형을 지정합니다. 이 옵션은 DRDA, SQLNET 및 NET8 랩퍼에 대한 필수사항입니다. 지원되는 데이터 소스 유형의 목록은 1403 페이지의 『부록F. 연합 시스템』에서 참조하십시오.

VERSION

*server-name*으로 표시된 데이터 소스의 버전을 지정합니다.

version

버전 번호를 지정합니다. *version*은 정수여야 합니다.

release

*version*으로 표시된 버전의 릴리스 번호를 지정합니다. *release*는 정수여야 합니다.

mod

*release*로 표시된 릴리스의 수정 번호를 지정합니다. *mod*는 정수여야 합니다.

version-string-constant

완전한 버전 지정을 합니다. *version-string-constant*는 단일 값(예: '8i') 이거나 *version*, *release* 및 *mod*의 결합된 값(예: '8.0.3')일 수 있습니다.

WRAPPER *wrapper-name*

server-type 및 *'server-version'*에 의해 표시된 유형 및 버전의 데이터 소스와 상호 작용하기 위해 연합 서버가 사용하는 랩퍼를 명명합니다.

AUTHORIZATION *remote-authorization-name*

CREATE SERVER문이 처리될 때 데이터 소스에서 필요한 조치가 수행되는 권한 부여 ID를 지정합니다. 이 ID는 필요한 조치가 요구하는 권한(BINDADD 또는 이에 상응하는 것)을 보유해야 합니다.

PASSWORD *password*

*remote-authorization-name*으로 표시되는 권한 부여 ID와 연관된 암호를 지정합니다. *password*가 지정되지 않았다면 사용자가 연합 사용자에 연결되어 있는 ID의 암호가 기본값이 됩니다.

OPTIONS

작동 가능화될 서버 옵션을 나타냅니다. *server-option-name*의 설명과 설정값은 1407 페이지의 『서버 옵션』에서 참조하십시오.

ADD

하나 이상의 서버 옵션을 작동가능화 시킵니다.

server-option-name

*server-name*으로 표시된 데이터 소스에 대한 정보를 구성하거나 제공하는 데 사용될 서버 옵션의 이름을 지정합니다.

string-constant

*server-option-name*에 대한 설정값을 문자열 상수로서 지정합니다.

주

- *remote-authorization-name*이 지정되지 않으면, 엽합 데이터베이스에 대한 권한 부여 ID가 사용될 것입니다.
- 데이터 소스가 필요로 하는 경우에는 *password*가 지정되어야 합니다. *password*에 있는 글자는 소문자여야 하며, *password*를 인용부호로 묶어야 합니다. *identifier*는 지정되고 *password*는 지정되지 않으면, *server-name*에 의해 표시된 데이터 소스의 인증 유형이 CLIENT로 간주됩니다.
- CREATE SERVER문을 사용하여 DB2 계열 인스턴스를 데이터 소스로 정의한 경우, DB2는 특정 패키지를 해당 인스턴스에 바인드해야 합니다. 바인드가 필요할 경우 명령문의 *remote-authorization-name*은 BIND 권한을 가져야 합니다. 바인드 완료에 걸리는 시간은 데이터 소스 속도와 네트워크 연결 속도에 따라 결정됩니다.

- 서버 옵션이 데이터 소스의 한 유형에 대해 어느 한 값으로 설정되고, 이 유형의 한 인스턴스에 대해 또 다른 값으로 설정되면, 인스턴스를 위해 두 번째 값이 첫번째 값을 겹쳐씁니다. 예를 들어, 연합 시스템의 OS/390용 DB2 Universal Database 데이터 소스에 대해 PASSWORD가 'Y'(데이터 소스에서 암호를 유효화함을 의미하는 yes)로 설정한다고 해 보십시오. 그런 후, 나중에 이 옵션의 기본값('N')에 SIBYL이라고 하는 특정 OS/390용 DB2 Universal Database 데이터 소스에 사용됩니다. 그 결과, SIBYL을 제외한 모든 OS/390용 DB2 Universal Database 데이터 소스에서 암호가 유효화될 것입니다.

예

예 1: 랩퍼 DB2WRAP를 통해 액세스 가능한 MVS/ESA용 DB2 4.1 데이터 소스를 정의하십시오. 데이터 소스 CRANDALL을 호출하고, 또한, 다음과 같이 지정하십시오.

- MURROW 및 DROWSSAP는 이 명령문이 처리될 때 CRANDALL에서 바인드되는 패키지하의 권한 부여 ID 및 암호가 될 것입니다.
- CRANDALL은 MYNODE라고 하는 인스턴스로서 DB2 RDBMS에 정의됩니다.
- 연합 서버가 CRANDALL을 액세스할 때, MYDB라는 데이터베이스로 연결될 것입니다.
- CRANDALL이 액세스될 수 있는 권한 부여 ID 및 암호가 대문자로 CRANDALL에 전송될 것입니다.
- MYDB 및 연합 데이터베이스는 동일한 조합 순서를 사용합니다.

```
CREATE SERVER CRANDALL
   TYPE DB2/MVS
   VERSION 4.1
   WRAPPER DB2WRAP
   AUTHORIZATION MURROW
   PASSWORD DROWSSAP
   OPTIONS
       ( NODE 'MYNODE',
       DBNAME 'MYDB',
       FOLD_ID 'U',
       FOLD_PW 'U',
       COLLATING_SEQUENCE 'Y' )
```

예 2: KLONDIKE라는 랩퍼를 통해 액세스 가능한 Oracle 7.2 데이터 소스를 정의합니다. 데이터 소스 CUSTOMERS를 호출하고, 다음과 같이 지정하십시오.

- CUSTOMERS는 ABC라는 인스턴스로서 Oracle RDBMS에 정의됩니다.

최적화 알고리즘을 위해 이들 통계를 제공하십시오.

- 연합 서버에 대한 CPU는 CUSTOMERS를 지원하는 CPU보다 두 배 빨리 수행합니다.
- 연합 서버에 있는 입출력 장치는 CUSTOMERS에 있는 입출력 장치보다 1.5 배 신속하게 데이터를 처리합니다.

```
CREATE SERVER CUSTOMERS
   TYPE ORACLE
   VERSION 7.2
   WRAPPER KLONDIKE
   OPTIONS
       ( NODE 'ABC',
       CPU_RATIO '2.0',
       IO_RATIO '1.5' )
```

CREATE TABLE

CREATE TABLE문은 테이블을 정의합니다. 정의에는 컬럼의 이름과 속성이 포함되어야 합니다. 정의에는 기본 키나 점검 제한조건과 같은 테이블의 다른 속성이 포함될 수도 있습니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 데이터베이스에 대한 CREATETAB 권한 및 테이블 공간에 대한 USE 특권은 다음 경우 중 하나입니다.
  - 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 테이블의 스키마 이름이 기존 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

  서브테이블이 정의되고 있는 경우, 권한 부여 ID는 테이블 계층의 루트 테이블의 정의자와 동일해야 합니다.

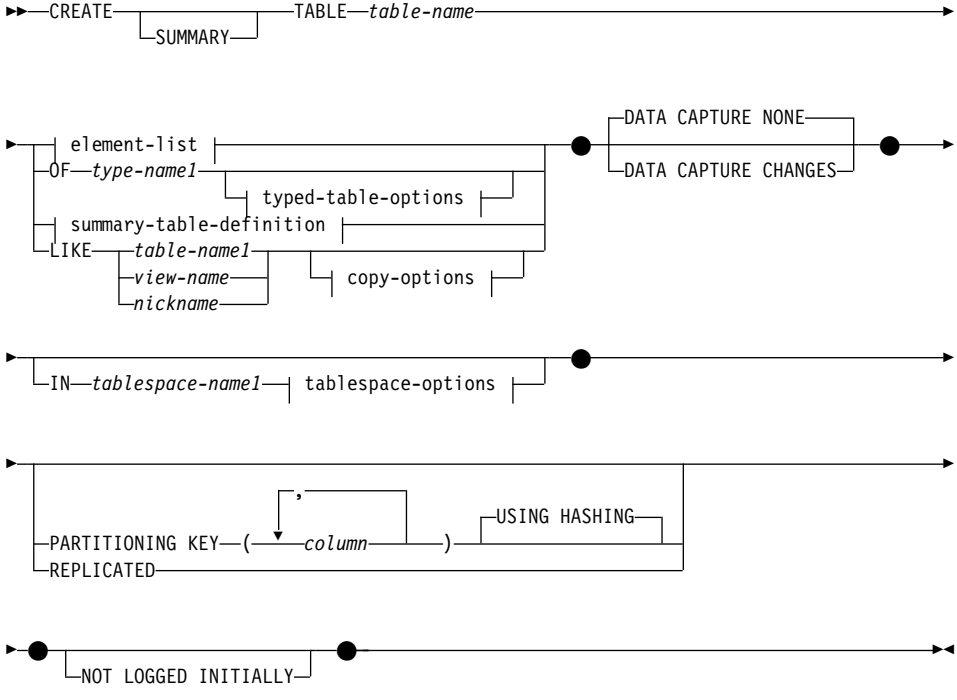
외부 키를 정의하려면, 명령문의 권한 부여 ID에 보유된 특권은 상위 테이블에서 다음 중 하나를 포함해야 합니다.

- 테이블에 대해 REFERENCES 특권
- 지정된 상위 키의 각 컬럼에 대한 REFERENCES 특권
- 테이블에 대해 CONTROL 특권
- SYSADM 또는 DBADM 권한

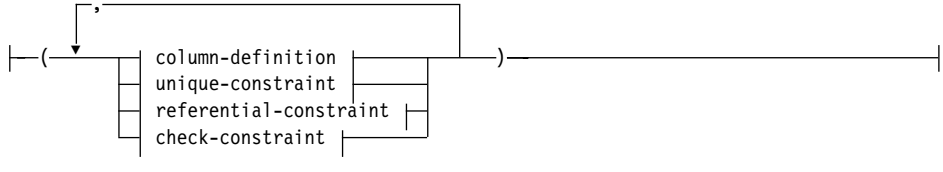
요약 테이블을 정의하려면(fullselect 사용), 명령문의 권한 부여 ID가 보유하는 특권에 fullselect에서 식별된 각 테이블이나 뷰에는 최소한 다음 중 하나가 포함되어야 합니다.

- REFRESH DEFERRED 또는 REFRESH IMMEDIATE가 지정된 경우, 테이블 또는 뷰에 대한 SELECT 특권 또는 ALTER 특권
- 테이블 또는 뷰에서의 CONTROL 특권
- SYSADM 또는 DBADM 권한

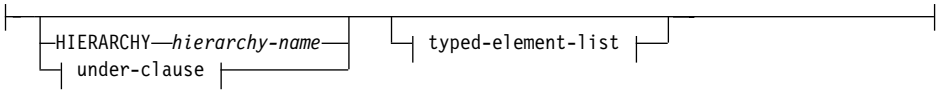
구문



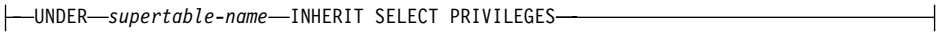
element-list:



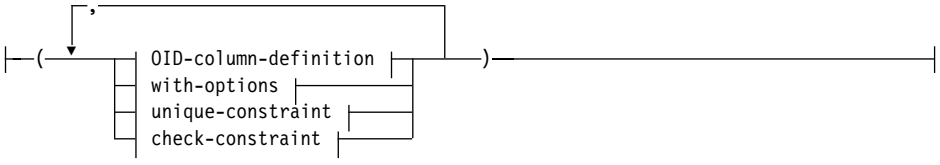
typed-table-options:



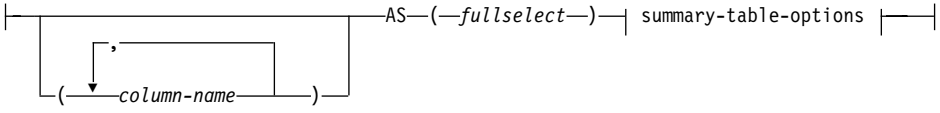
under-clause:



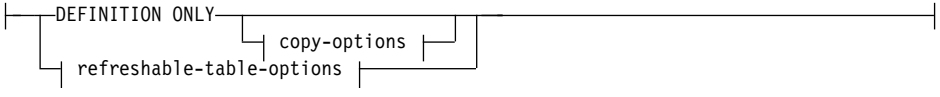
typed-element-list:



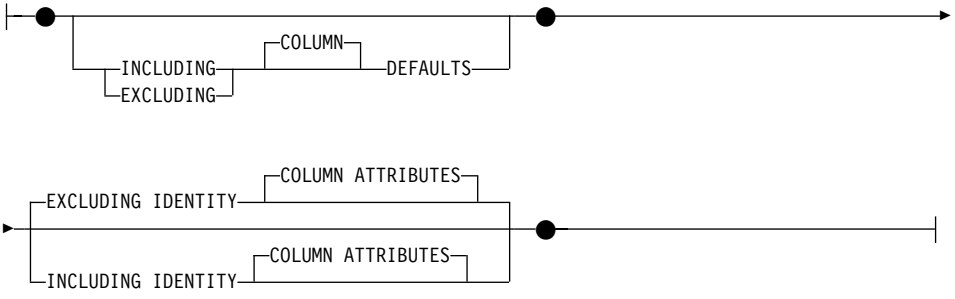
summary-table-definition:



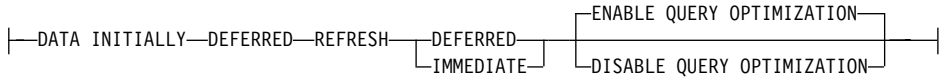
summary-table-options:



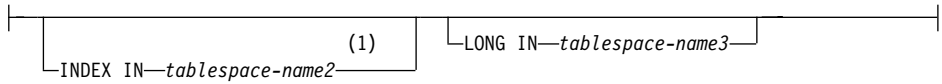
copy-options:



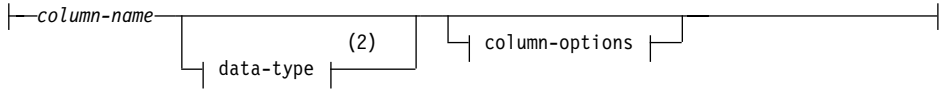
refreshable-table-options:



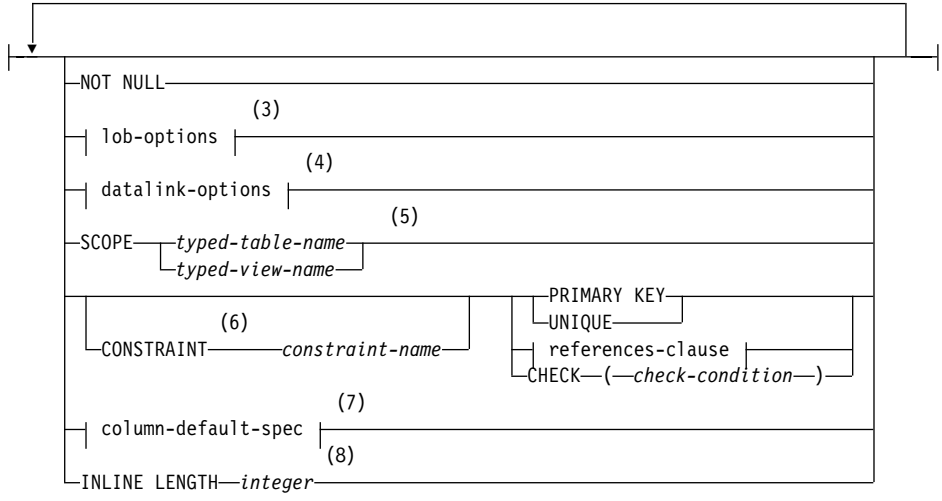
tablespace-options:



column-definition:



column-options:



주:

1 테이블이 작성되었을 때만 수행되는 테이블 색인용 테이블 공간 지정

2 선택된 첫번째 column-option이 생성 표현식을 갖는 column-default-spec인 경우에는 data-type을 생략할 수 있습니다. 이는 generation-expression의 결과 데이터 유형으로부터 결정됩니다.

3 lob 옵션 절은 대형 오브젝트 유형(BLOB, CLOB 및 DBCLOB)과 대형 오브젝트(LOB) 유형을 기초로 하는 구별 유형에만 적용됩니다.

4 데이터 링크 옵션 절은 DATALINK 유형과 DATALINK 유형에 기초한 구별 유형에만 적용됩니다. 이러한 유형에 있어서 LINKTYPE URL절은 필수 적입니다.

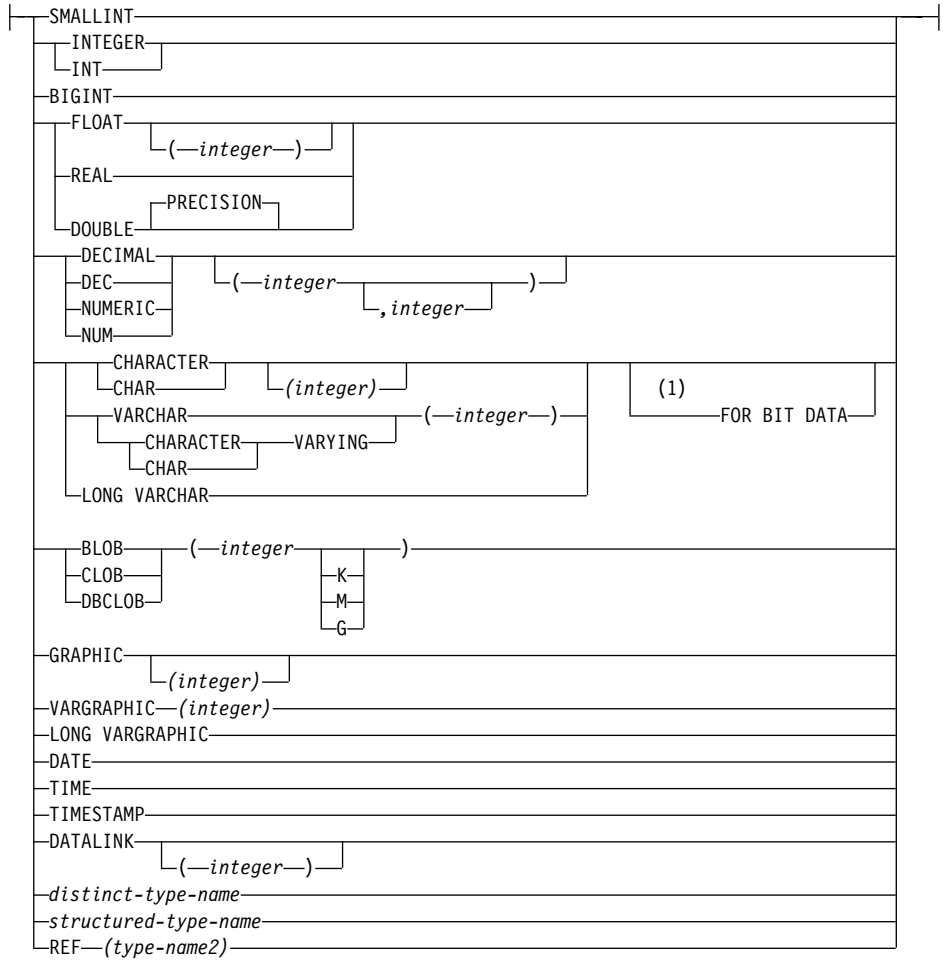
5 SCOPE절은 REF 유형에만 적용됩니다.

6 버전 1과의 호환성을 위해, CONSTRAINT 키워드는 참조 절을 정의하는 *컬럼 정의*에서 생략될 수 있습니다.

7 IDENTITY 컬럼 속성은 둘 이상의 파티션이 있는 EEE(Extended Enterprise Edition) 데이터베이스에서는 지원되지 않습니다.

8 INLINE LENGTH는 구조화 유형으로 정의된 컬럼에만 적용됩니다.

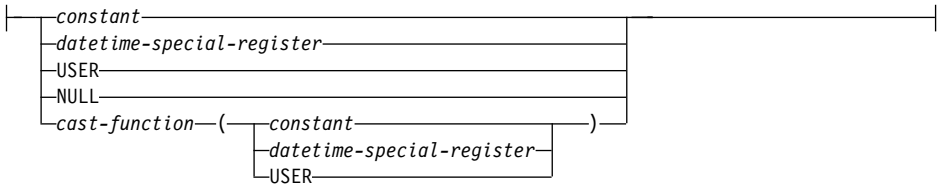
data-type:



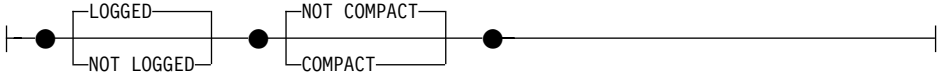
주:

1 FOR BIT DATA절은 다른 컬럼 제한조건에 대해 임의의 순서로 지정될 수 있습니다.

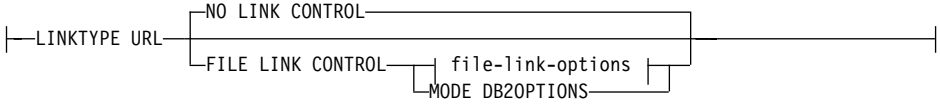
default-values:



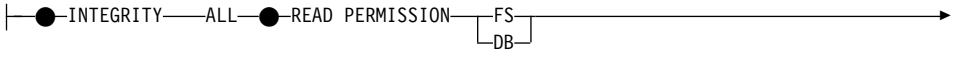
lob-options:

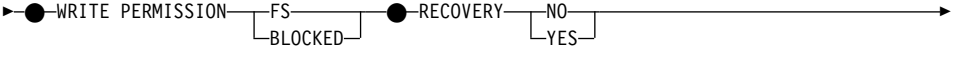


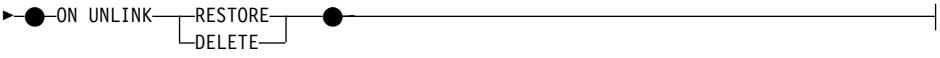
datalink-options:



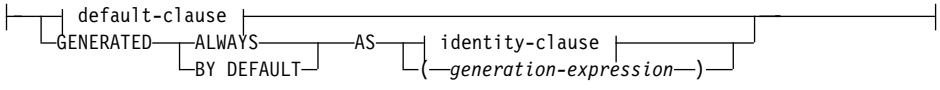
file-link-options:



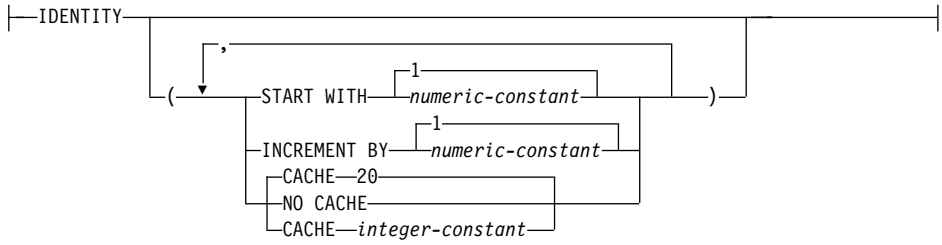




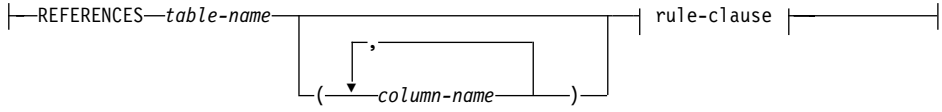
column-default-spec:



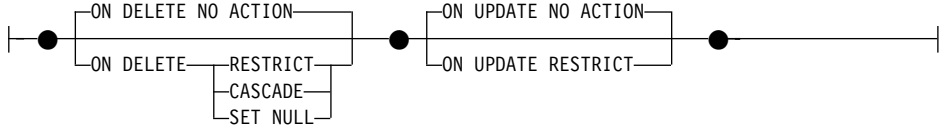
identity-clause:



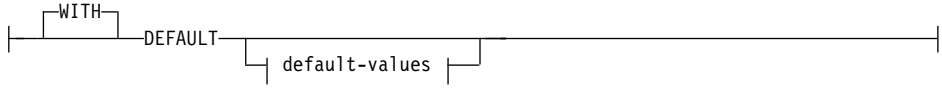
references-clause:



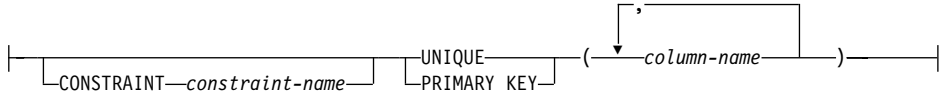
rule-clause:



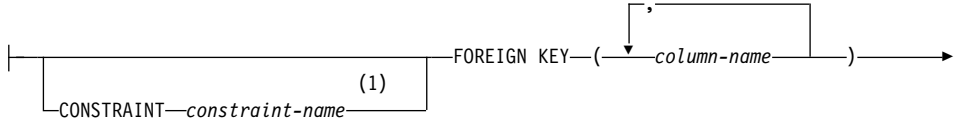
default-clause:

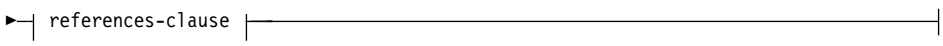


unique-constraint:

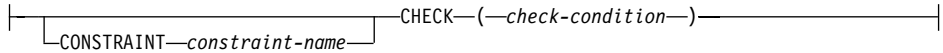


referential-constraint:

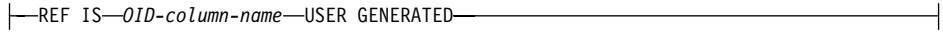




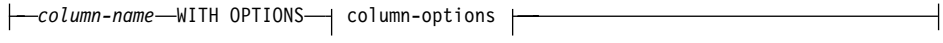
check-constraint:



OID-column-definition:



with-options:



주:

1 버전 1과의 호환성을 위해 제한조건 이름은 FOREIGN KEY 다음에 지정할 수도 있습니다(CONSTRAINT 키워드 없이).

설명

SUMMARY

요약 테이블이 정의 중임을 나타냅니다. 키워드는 선택적이지만, 지정될 때 명령문에 *summary-table-definition*이 포함됩니다(SQLSTATE 42601).

table-name

테이블을 명명합니다. 암시적 또는 명시적인 규정자를 포함하여, 이름은 카탈로그에 설명된 테이블, 뷰 또는 별명을 식별해서는 안 됩니다. 스키마 이름은 SYSIBM, SYSCAT, SYSFUN 또는 SYSSTAT이면 안 됩니다(SQLSTATE 42939).

OF *type-name1*

테이블의 컬럼이 *type-name1*에 의해 식별되는 구조화 유형 속성에 기초하도록 지정합니다. 스키마 이름 없이 *type-name1*이 지정되면, 유형 이름은 SQL 경로(정적 SQL인 경우에는 FUNCPATH 사전 처리 옵션, 동적 SQL인 경우에는 CURRENT PATH 레지스터에 의해 정의)에서 스키마를 검색하여 분석됩니다. 유형 이름은 기존 사용자 정의 유형의 이름이어야 하고(SQLSTATE

42704) 최소한 하나의 속성을 갖는(SQLSTATE 42997) 인스턴스 작성 가능한 구조화 유형이어야 합니다(SQLSTATE 428DP).

UNDER가 지정되지 않으면 오브젝트 식별자 컬럼을 지정해야 합니다(*OID-column-definition* 참조). 이 오브젝트 식별자 컬럼이 테이블의 첫번째 컬럼입니다. 오브젝트 ID 컬럼 뒤에는 *type-name1* 속성에 기초한 컬럼이 옵니다.

HIERARCHY *hierarchy-name*

테이블 계층과 연관된 계층 테이블에 이름을 부여합니다. 계층의 루트 테이블과 동시에 작성됩니다. 유형화된 테이블 계층에 있는 모든 서브테이블에 대한 데이터는 계층 테이블에 보관됩니다. 계층 테이블은 SQL문에서 직접 참조될 수 없습니다. *hierarchy-name*은 *table-name*입니다. 암시적 또는 명시적인 스키마 이름을 포함하여 *hierarchy-name*은 카탈로그에 테이블, 별명, 뷰 또는 별명을 식별해서는 안 됩니다. 스키마 이름이 지정된 경우, 작성되는 테이블의 스키마 이름과 동일해야 합니다. 루트 테이블 정의시 이 절을 생략하면, 식별자가 기존 테이블, 뷰, 별명 및 별명의 식별자 내에서 고유하도록 고유한 접미부가 뒤따르는 작성중인 테이블의 이름으로 이루어진 시스템에 의해 이름이 생성됩니다.

UNDER *supertable-name*

테이블이 *supertable-name*의 서브테이블임을 나타냅니다. 상위 테이블은 기존 테이블이어야 하고(SQLSTATE 42704), 이 테이블은 *type-name1*의 직접 상위 테이블인 구조화 유형을 사용하여 정의되어야 합니다(SQLSTATE 428DB). *table-name* 및 *supertable-name*은 동일해야 합니다(SQLSTATE 428DQ). *supertable-name*에 의해 식별되는 테이블에는 *type-name1*(SQLSTATE 42742)을 사용하여 이미 정의된 서브테이블을 없애야 합니다.

테이블 컬럼에는 REF(*type-name1*)로 수정될 유형을 가진 상위 유형의 오브젝트 식별자가 포함되며, 그 뒤에는 *type-name1* 속성에 기초한 컬럼들이 옵니다.(이 유형에는 상위 유형 속성이 포함된다는 점에 유의하십시오.) 속성 이름은 OID 컬럼 이름과 동일해서는 안 됩니다(SQLSTATE 42711).

테이블 공간, 데이터 캡쳐, 초기에 기록되지 않은 파티션 키 옵션과 같은 다른 테이블 옵션들은 지정될 수 없습니다. 이러한 옵션들은 상위 테이블로부터 물려받습니다(SQLSTATE 42613).

INHERIT SELECT PRIVILEGES

상위 테이블에 대한 SELECT 특권을 보유하고 있는 사용자나 그룹에게는 새로 작성된 서브테이블에 대해서도 그에 상응하는 특권이 권한 부여됩니다. 적절한 지정자가 이 특권의 권한을 준 사용자로 간주됩니다.

element-list

테이블의 요소를 정의합니다. 여기에는 테이블에 있는 컬럼 및 제한조건 정의가 포함됩니다.

typed-element-list

입력된 테이블의 추가 요소를 정의합니다. 여기에는 컬럼에 대한 추가 옵션, 오브젝트 식별자 컬럼 추가(루트 테이블만), 테이블에 대한 제한조건 등이 포함됩니다.

summary-table-definition

테이블 정의가 조회 결과를 기본으로 하면, 테이블은 조회를 기본으로 한 요약 테이블입니다.

column-name

테이블 내의 컬럼을 명명합니다. 컬럼 이름의 목록이 지정되면, 이 목록에 포함된 이름 수는 fullselect의 결과 테이블에 있는 컬럼 수와 동일해야 합니다. 각 *column-name*은 고유해야 하고 규정화되어서는 안 됩니다. 컬럼 이름의 목록이 지정되지 않으면, 테이블의 컬럼은 fullselect의 결과 컬럼에 대한 이름을 계승합니다.

fullselect의 결과 테이블에 명명되지 않은 컬럼의 복사 컬럼 이름이 있는 경우, 컬럼 이름 목록을 반드시 지정해야 합니다(SQLSTATE 42908). 명명되지 않은 컬럼은 컨테이너, 함수, 표현식 또는 세트 조작으로부터 유래된 컬럼으로서 선택 목록의 AS절을 사용하여 명명되지 않았습니다.

AS

테이블 정의 및 테이블에 포함된 데이터를 판별하는 데 사용되는 조회를 소개합니다.

fullselect

테이블이 기본으로 하는 조회를 정의합니다. 결과 컬럼 정의는 동일한 조회로 정의된 뷰에 대한 것들과 동일합니다.

모든 선택 목록 요소에는 이름이 있어야 합니다(표현식의 AS절 사용에 대해서는 435 페이지의 『select절』에서 참조하십시오). 지정된 *summary-table-options*는 요약 테이블의 속성을 정의합니다. 선택된 옵션은 다음과 같이 fullselect의 내용도 정의합니다.

DEFINITION ONLY가 지정될 때에는 유형화된 테이블이나 유형화된 뷰를 참조하지 않는 유효한 fullselect만 지정될 수 있습니다.

REFRESH DEFERRED 또는 REFRESH IMMEDIATE가 지정될 때, fullselect는 다음을 포함할 수 없습니다(SQLSTATE 428EC).

- FROM 절에서 별명, 요약 테이블, 선언된 임시 테이블 또는 유형화 테이블에 대한 참조
- 뷰의 fullselect가 요약 테이블의 fullselect에서 나열된 제한사항을 위반하는 뷰에 대한 참조
- 참조 유형 또는 DATALINK 유형인 표현식(또한 이러한 유형을 기본으로 한 구별 유형)
- 외부 조치를 갖는 함수
- SQL에 쓰여진 함수
- 물리적 특성에 의존하는 함수(예: NODENUMBER, PARTITION)
- 시스템 오브젝트에 테이블 또는 뷰 참조(explain 테이블도 지정되어 있지 않아야 함)
- 구조화 유형이나 LOB 유형(또는 LOB 유형에 기초하는 구별 유형)인 표현식

REFRESH IMMEDIATE이 지정될 때

- fullselect가 부속 선택이어야 합니다.
- 부속 선택은 다음을 포함할 수 있습니다.
  - 결정할 수 없는 함수
  - 스칼라 fullselects
  - fullselects을 가진 술어
  - 특수 레지스터

- 요약 테이블이 복제되지 않는한 GROUP BY 절은 부속 선택에 포함되어야 합니다.
- 지원된 컬럼 함수는 SUM, COUNT, COUNT_BIG 및 GROUPING(DISTINCT 없이)입니다. 선택 목록은 COUNT(*) 또는 COUNT_BIG(*) 컬럼을 포함해야 합니다. 요약 테이블 선택 목록이 SUM(X)를 포함하면(여기서 X는 널(NULL) 입력 가능 인수), 요약 테이블이 선택 목록에 COUNT(X)도 가지고 있어야 합니다. 이들 컬럼 함수는 표현식의 일부가 될 수 없습니다.
- FROM절이 두 개 이상의 테이블이나 뷰를 참조하면, 명시적 INNER JOIN 구문을 사용하지 않고 내부 조인을 정의할 수만 있습니다.
- 모든 GROUP BY 항목이 선택 목록에 포함되어야 합니다.
- GROUPING SETS, CUBE 및 ROLLUP이 지원됩니다. 선택 목록에 있는 GROUP BY 항목 및 관련 GROUPING 컬럼 기능은 결과 세트의 고유 키를 형성합니다. 이와 같이 하려면, 다음 제한사항을 충족시켜야 합니다.
  - 집합 그룹화를 반복할 수 없습니다. 예를 들어, `ROLLUP(X,Y), X`는 허용되지 않는데, 그 이유는 `GROUPING SETS((X,Y),(X),(X))`와 동등하지 않기 때문입니다.
  - X가 GROUPING SETS, CUBE 또는 ROLLUP내에 나타나는 널(NULL) 입력 가능 GROUP BY 항목인 경우, GROUPING(X)가 선택 목록에 나타나야 합니다.
  - 상수에 그룹화가 허용되지 않습니다.
- HAVING절이 허용되지 않습니다.
- 복수 파티션 노드 그룹에서는 파티션 키가 항목별 그룹의 부속 집합이거나 요약 테이블을 복제해야 합니다.

summary-table-options

요약 테이블의 속성을 정의합니다.

DEFINITION ONLY

조회가 테이블을 정의하는 데에만 사용됩니다. 테이블은 조회 결과를 사용

하여 생성되지 않으며 REFRESH TABLE문은 사용될 수 없습니다. CREATE TABLE문이 완료되면, 테이블은 더 이상 요약 테이블로 간주되지 않습니다.

테이블의 컬럼은 fullselect의 결과인 컬럼의 정의에 기초하여 정의됩니다. fullselect가 FROM 절에서 하나의 테이블을 참조하는 경우, 해당 테이블의 컬럼인 선택 목록 항목은 컬럼 이름, 데이터 유형 및 참조된 테이블의 널(NULL) 입력 가능 특성을 사용하여 정의됩니다.

refreshable-table-options

요약 테이블 속성의 옵션을 새로 고칠 수 있도록 정의합니다.

DATA INITIALLY DEFERRED

데이터가 CREATE TABLE문의 일부로서 테이블에 삽입되지 않습니다. *table-name*을 지정하는 REFRESH TABLE문은 데이터를 테이블에 삽입하는 데 사용됩니다.

REFRESH

테이블의 데이터가 유지보수되는 방법을 나타냅니다.

DEFERRED

테이블의 데이터가 REFRESH TABLE문을 사용하여 언제라도 화면갱신될 수 있습니다. 테이블의 데이터는 REFRESH TABLE문이 처리될 때의 스냅샷인 조회 결과에만 영향을 미칩니다. 이 속성이 정의되어 있는 요약 테이블에서는 NSERT, UPDATE 또는 DELETE문을 사용할 수 없습니다(SQLSTATE 42807).

IMMEDIATE

DELETE, INSERT 또는 UPDATE의 일부로서 기저 테이블에 이루어진 변경사항은 요약 테이블에 연쇄됩니다. 이런 경우, 테이블의 내용은 지정된 *부속 선택*이 처리될 경우와 항상 동일합니다. 이 속성이 정의되어 있는 요약 테이블에서는 INSERT, UPDATE 또는 DELETE문을 사용할 수 없습니다(SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION

적절한 환경에서 조회 최적화를 위해 요약 테이블을 사용할 수 있습니다.

DISABLE QUERY OPTIMIZATION

요약 테이블을 조회 최적화에 대해 사용할 수 없습니다. 테이블이 아직 직접적으로 조회 중에 있습니다.

LIKE *table-name1* **또는** *view-name* **또는** *nickname*

테이블 컬럼에 식별된 테이블(*table-name1*), 뷰(*view-name*) 또는 별명(*nickname*)의 컬럼과 정확히 같은 이름과 설명이 있음을 지정합니다. LIKE 다음에 지정된 이름은 카탈로그 또는 선언된 임시 테이블에 있는 테이블, 뷰 또는 별명을 식별해야 합니다. 입력된 테이블 또는 입력된 뷰는 지정할 수 없습니다(SQLSTATE 428EC).

LIKE의 사용은 *n* 컬럼의 내재된 정의입니다. 여기서 *n*은 식별된 테이블, 뷰 또는 별명의 컬럼 수입니다.

- 식별된 테이블인 경우 내재된 정의에는 *table-name1*의 각 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 입력 가능 특성이 포함됩니다. EXCLUDING COLUMN DEFAULTS가 지정되지 않을 경우에는 컬럼 기본값도 포함됩니다.
- 식별된 뷰인 경우 내재된 정의에는 *view-name*에 정의된 fullselect의 각 결과 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(NULL) 입력 가능 특성이 포함됩니다.
- 식별된 별명인 경우 내재된 정의에는 *nickname*의 각 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(NULL) 입력 가능 특성이 포함됩니다.

복사 속성 절을 기초로 컬럼 기본값 및 식별 컬럼 속성을 포함하거나 제외시킬 수 있습니다. 내재된 정의에는 식별된 테이블, 뷰 또는 별명의 다른 어느 속성도 포함되지 않습니다. 그러므로 새로운 테이블이 고유한 제한조건, 외부키 제한조건, 트리거 또는 색인을 가지지 않습니다. 테이블은 IN절에 의해 내재적 또는 명시적으로 지정된 테이블 공간에서 작성되며, 선택적 절이 지정되는 경우에만 테이블이 다른 선택적 절을 가집니다.

copy-options

이러한 옵션은 소스 결과 테이블 정의(테이블, 뷰 또는 fullselect)의 추가 속성을 복사할지 여부를 지정합니다.

INCLUDING COLUMN DEFAULTS

소스 결과 테이블 정의의 갱신 가능 컬럼 각각에 대한 컬럼 기본값이 복사됩니다. 갱신 가능하지 않은 컬럼은 작성된 테이블의 해당 컬럼에 정의된 기본값을 가지지 않게 됩니다.

LIKE *table-name*이 지정되고 *table-name*이 기본 테이블 또는 선언된 임시 테이블을 식별하는 경우에는 INCLUDING COLUMN DEFAULTS가 기본값입니다.

EXCLUDING COLUMN DEFAULTS

컬럼 기본값은 소스 결과 테이블 정의로부터 복사되지 않습니다.

LIKE *table-name*이 지정되고 *table-name*이 기본 테이블 또는 선언된 임시 테이블을 식별하는 경우를 제외하고는 이 절이 기본값입니다.

INCLUDING IDENTITY COLUMN ATTRIBUTES

가능한 경우 식별 컬럼 속성(START WITH, INCREMENT BY 및 CACHE 값)이 소스 결과 테이블 정의로부터 복사됩니다. 테이블, 뷰 또는 fullselect의 해당 컬럼 요소가 테이블 컬럼의 이름이거나 직접 또는 간접으로 식별 등록 정보를 가지는 기본 테이블 컬럼의 이름에 해당되는 뷰 컬럼의 이름인 경우 식별 컬럼 속성을 복사할 수 있습니다. 다른 모든 경우에는 새 테이블의 컬럼이 식별 등록 정보를 가져오지 않습니다. 예를 들면,

- fullselect의 선택 목록에는 식별 컬럼 이름의 여러 인스턴스가 포함됩니다(즉, 두 번 이상 같은 컬럼 선택).
- fullselect의 선택 목록에 여러 식별 컬럼이 포함됩니다(즉, 하나의 조인 포함).
- 식별 컬럼이 선택 목록의 표현식에 포함됩니다.
- fullselect에 하나의 설정 연산(union, except 또는 intersect)이 포함됩니다.

EXCLUDING IDENTITY COLUMN ATTRIBUTES

식별 컬럼 속성이 소스 결과 테이블 정의로부터 복사되지 않습니다.

column-definition

컬럼의 속성을 정의합니다.

column-name

테이블의 컬럼을 명명합니다. 이름을 규정화할 수 없으며, 테이블의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다.

테이블은 다음 사항을 포함합니다.

- 최대 500 컬럼이 있는 4K 페이지 *크기*(여기서, 컬럼의 바이트 수는 4K 페이지 *크기* 단위로 4005보다 커서는 안 됩니다). 853 페이지의 『행 크기』에서 자세한 내용을 참조하십시오.
- 최대 1 012 컬럼을 가진 8K 페이지 *크기*(여기서 컬럼의 바이트 합계가 8081을 초과해서는 안 됩니다). 853 페이지의 『행 *크기*』에서 자세한 내용을 참조하십시오.
- 최대 1 012 컬럼을 가진 16K 페이지 *크기*(여기서, 컬럼의 바이트 수는 16 293 보다 커서는 안 됩니다).
- 최대 1 012 컬럼을 가진 32K 페이지 *크기*(여기서, 컬럼의 바이트 수는 32 677 보다 커서는 안 됩니다).

data-type

다음 목록에 있는 유형 중 하나입니다. 다음을 이용하십시오.

SMALLINT

작은 정수의 경우.

INTEGER 또는 **INT**

큰 정수의 경우

BIGINT

대 정수(big integer)의 경우.

FLOAT(*integer***)**

단정밀도 또는 배정밀도의 부동 소수점 수의 경우, *integer*의 값에 따릅니다. 정수의 값은 1 - 53이 범위 내에 있어야 합니다. 1 - 24 사이의 값은 단일 정밀도를 나타내며 25 - 53 사이의 값은 배정밀도를 나타냅니다.

또한, 다음을 지정할 수 있습니다.

REAL	단정밀도 부동 소수점.
DOUBLE	배정밀도 부동 소수점

DOUBLE PRECISION	배정밀도 부동 소수점
FLOAT	배정밀도 부동 소수점

DECIMAL*(precision-integer, scale-integer)* 또는 **DEC***(precision-integer, scale-integer)*

십진수의 경우. 첫번째 정수는 숫자의 정밀도, 즉 총 자리수로, 그 범위는 1 - 31이 될 수 있습니다. 두 번째 정수는 숫자의 스케일, 즉 소수점의 오른쪽에 있는 자리수로, 그 범위는 0 - 숫자의 정밀도가 될 수 있습니다.

정밀도나 스케일을 지정하지 않으면, 5,0의 기본값이 사용됩니다. **NUMERIC**와 **NUM**는 **DECIMAL**과 **DEC**의 동의어로 사용될 수 있습니다.

CHARACTER*(integer)* 또는 **CHAR***(integer)* 또는 **CHARACTER** 또는 **CHAR**

범위가 1 - 254가 될 수 있는, *integer* 길이의 고정 길이 문자열. 길이를 지정하지 않으면, 1자로 간주됩니다.

VARCHAR*(integer)***,** 또는 **CHARACTER VARYING***(integer)***,** 또는 **CHAR VARYING***(integer)*

1 - 32 672 범위에 있을 수 있는 최대 길이 *integer*의 가변 길이 문자열의 경우.

LONG VARCHAR

최대 길이가 32700인 가변 길이 문자열.

FOR BIT DATA

비트(2진) 데이터로 취급되는 컬럼의 내용을 지정합니다. 다른 시스템과의 데이터 교환중에는 코드 페이지 변환이 수행되지 않습니다. 비교는 데이터베이스 배열 순서와 상관없이 2진으로 수행됩니다.

BLOB*(integer [K | M | G])*

지정된 길이(바이트 단위)의 2진 대형 오브젝트(BLOB) 문자열.

길이의 범위는 1바이트 - 2 147 483 647바이트가 될 수 있습니다.

integer 그 자체로 지정된 경우, 그것은 최대 길이입니다.

integer K(대문자 또는 소문자)를 지정한 경우, 최대 길이는 1 024 *integer*입니다. *integer*의 최대값은 2 097 152입니다.

*integer M*을 지정하면, 최대 길이는 1 048 576 x *integer*입니다. *integer*의 최대값은 2 048입니다.

*integer G*를 지정하면, 최대 길이는 1 073 741 824 x *integer*입니다. *integer*의 최대값은 2입니다.

1기가바이트 이상의 BLOB 문자열을 작성하려면, NOT LOGGED 옵션을 지정해야 합니다.

정수와 K, M 또는 G 사이의 어떤 수든 허용됩니다. 공백은 필요없습니다. 예를 들어, 다음은 모두 유효합니다.

```
BLOB(50K)    BLOB(50 K)    BLOB (50   K)
```

CLOB*(integer [K | M | G])*[74]

지정된 최대 길이(바이트 단위)의 대형 오브젝트(LOB) 문자열.

*정수 K | M | G*의 의미는 BLOB와 유사합니다.

1기가바이트 이상의 CLOB 문자열을 작성하려면, NOT LOGGED 옵션을 지정해야 합니다.

DBCLOB*(integer [K | M | G])*

지정된 최대 길이(2바이트 문자열)의 2바이트 대형 오브젝트 문자열.

*정수 K | M | G*의 의미는 BLOB와 유사합니다. 차이점은 지정된 숫자가 2바이트 문자의 수인 것과 최대 크기가 1 073 741 823개의 2바이트 문자라는 것입니다.

1 기가바이트 이상의 DBCLOB 문자열을 지정하려면 NOT LOGGED 옵션을 지정해야 합니다.

GRAPHIC*(integer)*

범위가 1 - 127이 될 수 있는, *integer* 길이의 고정 길이 그래픽 문자열. 길이를 지정하지 않으면, 1자로 간주됩니다.

---

74. CLOB 컬럼에 FOR BIT DATA절을 지정할 수 없습니다. 그러나, CHAR FOR BIT DATA 문자열은 CLOB 컬럼에 지정할 수 있으며 CHAR FOR BIT DATA 문자열을 CLOB 문자열과 연결할 수 있습니다.

VARGRAPHIC*(integer)*

1 - 16 336 범위에 있을 수 있는 최대 길이 *integer*의 가변 길이 그래픽 문자열의 경우.

LONG VARGRAPHIC

최대 길이가 16 350인 가변 길이 그래픽 문자열.

DATE

날짜

TIME

시간

TIMESTAMP

시간소인

DATALINK 또는 **DATALINK***(integer)*

데이터베이스 외부에 저장된 데이터로 링크할 경우.

테이블에 있는 컬럼은 선택적 주석 뿐만 아니라 외부 데이터로의 링크를 설정 및 유지보수하는 데 필요한 참조 정보가 들어 있는 "앵커 값"으로 구성됩니다.

DATALINK 컬럼의 길이는 200바이트입니다. *integer*가 지정되면, 200 이어야 합니다. 길이를 지정하지 않으면 200자로 간주됩니다.

DATALINK 값은 내장된 스칼라 함수 세트를 가진 캡슐화 값입니다. DATALINK 값을 작성하기 위해 DLVALUE라는 함수가 있습니다. 다음 함수를 사용하여 DATALINK 값으로부터 속성을 추출할 수 있습니다.

- DLCOMMENT
- DLLINKTYPE
- DLURLCOMPLETE
- DLURLPATH
- DLURLPATHONLY
- DLURLSCHEME
- DLURLSERVER

DATALINK 컬럼에는 다음과 같은 제한사항이 있습니다.

- 컬럼은 색인의 일부가 될 수 없습니다. 따라서, 기본 키나 고유 제한조건 컬럼으로서 포함될 수 없습니다(SQLSTATE 42962).
- 컬럼은 참조 제한조건의 외부 키가 될 수 없습니다(SQLSTATE 42830).
- 컬럼에 대해 기본값(WITH DEFAULT)이 지정될 수 없습니다. 컬럼이 널(NULL) 입력 가능한 경우, 그 컬럼에 대한 기본값은 널(NULL)입니다(SQLSTATE 42894).

distinct-type-name

구별 유형인 사용자 정의 유형의 경우. 구별 유형 이름이 스키마 이름 없이 지정되면, 구별 유형 이름은 SQL 경로(정적 SQL의 경우 FUNCPATH 사전 처리 옵션에 의해 정의되고, 동적 SQL의 경우 CURRENT PATH 레지스터에 의해 지정)의 스키마를 검색하여 분석됩니다.

컬럼이 구별 유형을 사용하여 정의되면, 그 컬럼의 데이터 유형은 구별 유형입니다. 컬럼의 길이와 스케일은 각각 구별 유형의 길이와 원시 유형의 스케일입니다.

구별 유형을 사용하여 정의된 컬럼이 참조 제한조건의 외부 키이면, 1차 키의 해당 컬럼에 대한 데이터 유형이 같은 구별 유형을 갖고 있어야 합니다.

structured-type-name

구조화 유형인 사용자 정의 유형의 경우. 스키마 이름 없이 구조화 유형 이름이 지정되면, 구조화 유형 이름은 SQL 경로(정적 SQL인 경우에는 FUNCPATH 선행 처리 옵션, 동적 SQL인 경우에는 CURRENT PATH 레지스터에 의해 정의)에서 스키마를 검색하여 분석됩니다.

컬럼이 구조화 유형을 사용하여 정의되면, 컬럼의 정적 데이터 유형은 구조화 유형입니다. 컬럼은 *structured-type-name*의 부속 유형인 동적 유형 값을 포함할 수 있습니다.

구조화 유형을 사용하여 정의된 컬럼은 기본 키, 고유 제한조건, 외부 키, 색인 키 또는 파티션 키에 사용할 수 없습니다(SQLSTATE 42962).

컬럼이 구조화 유형을 사용하여 정의되고 중첩 레벨의 참조 유형 속성을 포함하는 경우, 해당 참조 유형 속성이 범위에서 벗어납니다. 참조 해제 조작에 그러한 속성을 사용하려면, CAST 스펙을 사용하여 명시적으로 SCOPE를 지정해야 합니다.

DATALINK 유형 속성을 갖는 구조화 유형 또는 DATALINK를 기초로 구별 유형을 사용하여 컬럼을 정의한 경우, 이 컬럼은 널(NULL)이어야만 합니다. 이 유형의 구성자(constructor) 함수를 사용하면 오류가 리턴되므로(SQLSTATE 428ED) 이 유형의 인스턴스를 해당 컬럼에 삽입할 수 없습니다.

REF (*type-name2***)**

입력된 테이블에 대한 참조. 스키마 이름 없이 *type-name2*가 지정되면, 유형 이름은 SQL 경로(정적 SQL인 경우에는 FUNCPATH 사전 처리 옵션, 동적 SQL인 경우에는 CURRENT PATH 레지스터에 의해 정의)에서 스키마를 검색하여 분석됩니다. 컬럼의 기본 데이터 유형은 *type-name2*에 대한 CREATE TYPE문의 REF USING절에 지정된 프리젠테이션 데이터 유형이나 *type-name2*를 포함하는 데이터 유형 계층의 루트 유형을 기초로 합니다.

column-options

테이블 컬럼에 관련된 추가 옵션을 정의합니다.

NOT NULL

컬럼에 널(NULL)값이 포함되지 못하도록 합니다.

NOT NULL이 지정되지 않으면, 컬럼에 널(NULL) 값이 포함될 수 있고, 기본값은 널(NULL) 값이거나 WITH DEFAULT절에서 제공되는 값입니다.

lob-options

LOB 데이터 유형에 대해 옵션을 지정합니다.

LOGGED

컬럼에 대한 변경사항이 로그에 기록됨을 지정합니다. 그러한 컬럼의 데이터는 데이터베이스 유틸리티(RESTORE DATABASE와 같은)를 이용하여 복구 가능합니다. LOGGED가 기본값입니다.

1기가바이트 보다 큰 LOB은 기록될 수 없고(SQLSTATE 42993) 10 메가바이트 보다 큰 LOB은 기록해서는 안 됩니다.

NOT LOGGED

컬럼에 대한 변경사항이 로그에 기록되지 않음을 지정합니다.

NOT LOGGED는 확약 또는 구간 복원 조작에 영향을 주지 않습니다. 즉, 트랜잭션이 구간 복원되는 경우에도 LOB 값의 기록 여부에 관계없이 데이터베이스의 일관성이 유지됩니다. 기록하지 않는 것은 백업이나 적재 작업 다음에 롤 포워드 작업을 수행하는 동안 LOB 데이터가 롤 포워드 도중에 대체된 로그 레코드를 갖고 있었던 LOB 값에 대해 0으로 대체됨을 암시합니다. 복구시, 모든 확약된 변경사항과 구간 복원된 변경사항은 예상되는 결과를 반영합니다. 로그되지 않는 LOB 컬럼에 대해서는 *관리 안내서*에서 참조하십시오.

COMPACT

후속 추가 조작이 용이하도록 LOB 저장영역 끝에 공간을 남겨두는 대신 LOB 컬럼의 값이 최소의 디스크 공간을 차지하도록 지정합니다(LOB 값에 사용되는 마지막 그룹의 추가 디스크 페이지를 해제함). 이 방법으로 데이터를 저장하면 컬럼에 연산이 추가될 때(길이가 증가될 때) 성능이 저하됩니다.

NOT COMPACT

컬럼에서 LOB 값을 쉽게 변경할 수 있도록 삽입용으로 약간의 공간을 지정합니다. 이것은 기본값입니다.

datalink-options

DATALINK 데이터 유형과 연관된 옵션을 지정합니다.

LINKTYPE URL

이는 URL(Uniform Resource Locator: 단일 자원 위치 지정자)로서 링크 유형을 지정합니다.

NO LINK CONTROL

파일이 존재하는지 판별하는 확인 작업이 없도록 지정합니다. URL 구문만이 확인됩니다. 파일에 대한 데이터베이스 관리 프로그램 제어는 없습니다.

FILE LINK CONTROL

파일 존재에 대한 확인이 이루어지도록 지정합니다. 파일에 대한 데이터베이스 관리 프로그램의 추가 제어권을 제공하기 위해 추가 옵션을 사용할 수도 있습니다.

file-link-options

파일 링크의 데이터베이스 관리 프로그램 제어 레벨을 지정하기 위한 추가 옵션.

INTEGRITY

DATALINK 값과 실제 파일 사이의 링크 무결성 레벨을 지정합니다.

ALL

DATALINK 값으로 지정된 파일은 데이터베이스 관리 프로그램의 제어하에 있으므로, 표준 파일 시스템 프로그래밍 인터페이스로 삭제하거나 이름을 바꿀 수 없습니다.

READ PERMISSION

DATALINK 값에 지정된 파일을 읽기 위한 권한 사용 방식을 지정합니다.

FS

읽기 액세스 사용 권한은 파일 시스템 사용 권한에 의해 결정됩니다. 이런 파일들은 컬럼에서 파일 이름을 검색하지 않고도 액세스할 수 있습니다.

DB

읽기 액세스 사용 권한은 데이터베이스에 의해 결정됩니

다. 파일에 대한 액세스는 유효한 파일 액세스 토큰을 전달할 때만 허용되고, 열기 조작시 테이블로부터 DATALINK 값을 검색할 때 리턴됩니다.

WRITE PERMISSION

DATALINK 값에 지정된 파일에 쓰기 위한 권한 정의 방식을 지정합니다.

FS

쓰기 액세스 사용 권한은 파일 시스템 사용 권한에 의해 결정됩니다. 이런 파일들은 컬럼에서 파일 이름을 검색하지 않고도 액세스할 수 있습니다.

BLOCKED

쓰기 액세스가 블록되어 있습니다. 인터페이스를 통해서는 파일을 직접 갱신할 수 없습니다. 대체 메카니즘를 사용하여 정보를 갱신해야 합니다. 예를 들어, 파일이 복사되고, 복사본이 갱신된 후, 갱신된 DATALINK 값은 새로운 파일 사본을 포인트합니다.

RECOVERY

DB2가 이 컬럼에서 값에 의해 참조되는 파일의 시간 복구 시점을 지원할 것인지 지정합니다.

YES

DB2는 이 컬럼에서 값에 의해 참조되는 파일의 시간 복구 시점을 지원합니다. 이 값은 INTEGRITY ALL 및 WRITE PERMISSION BLOCKED도 지정할 때에만 지정할 수 있습니다.

NO

시간 복구 시점이 지원되지 않도록 지정합니다.

ON UNLINK

DATALINK 값이 변경되거나 삭제(링크가 해제)될 때 파일에서의 조치를 지정합니다. WRITE PERMISSION FS가 사용될 때는 적용할 수 없습니다.

RESTORE

파일 링크가 해제될 때, DataLink 파일 관리 프로그램이 파일이 링크되었을 때 있었던 사용 권한을 가지고 파일을 소유자에게 리턴하도록 지정합니다. 사용자가 더 이상 파일 서버에 등록되어 있지 않은 경우, 그 결과는 제품에 따라 달라집니다.[75] 이것은 INTEGRITY ALL과 WRITE PERMISSION BLOCKED도 지정된 경우에만 지정할 수 있습니다.

DELETE

파일 링크가 해제될 때 파일도 삭제되도록 지정합니다. 이는 READ PERMISSION DB 및 WRITE PERMISSION BLOCKED도 지정되었을 때에만 지정될 수 있습니다.

MODE DB2OPTIONS

이 모드는 기본 파일 링크 옵션 세트를 정의합니다. 기본값은 다음 DB2OPTIONS에 의해 정의됩니다.

- INTEGRITY ALL
- READ PERMISSION FS
- WRITE PERMISSION FS
- RECOVERY NO

ON UNLINK는 WRITE PERMISSION FS가 사용되므로 적용될 수 없습니다.

SCOPE

참조 유형 컬럼 영역을 식별합니다.

참조 해제(dereference) 연산자의 왼쪽 피연산자 또는 DEREF 함수의 인수로서 사용될 예정인 컬럼에 대한 영역이 지정되어야 합니다. 일

75. DB2 Universal Database로 파일은 특수 사전정의된 "dfmunknown" 사용자 ID에 지정됩니다.

반적으로 상호 참조 테이블인 경우, 참조 유형 컬럼 영역을 지정하면 이후의 ALTER TABLE문을 따르게 되어 목표 테이블이 정의될 수 있도록 합니다.

typed-table-name

입력된 테이블의 이름. 이미 테이블이 있어야 하고, 그렇지 않은 경우 작성되는 테이블 이름과 동일해야 합니다(SQLSTATE 42704). *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *typed-table-name* 유형입니다(SQLSTATE 428DM). 값이 실제로 *typed-table-name*에 있는 기존 행을 참조하도록 보장하기 위해 *column-name*에 지정된 값에 대해 어떠한 점검도 행해지지 않습니다.

typed-view-name

입력된 뷰의 이름. 이미 뷰가 있어야 하고, 그렇지 않은 경우 작성되는 뷰 이름과 동일해야 합니다(SQLSTATE 42704). *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *typed-view-name* 유형입니다(SQLSTATE 428DM). 값이 실제로 *typed-table-name*에 있는 기존 행을 참조하도록 보장하기 위해 *column-name*에 지정된 값에 대해 어떠한 점검도 행해지지 않습니다.

CONSTRAINT *constraint-name*

제한조건을 명명합니다. *constraint-name*은 같은 CREATE TABLE 문 내에 이미 지정된 제한조건을 식별해서는 안 됩니다(SQLSTATE 42710).

이 절이 생략되면, 테이블에 정의된 기존 제한조건의 식별자내에 있는 고유한 18자 식별자가 생성됩니다. [76]

PRIMARY KEY 또는 UNIQUE 제한조건과 함께 사용되는 경우, 제한조건을 지원하기 위해 작성된 색인의 이름으로 *constraint-name*을 사용할 수 있습니다.

---

76. 이 식별자는 뒤에 시간소인 함수로 생성된 15 숫자 문자가 오는 "SQL" 양식입니다.

PRIMARY KEY

이는 단일 컬럼으로 구성되는 기본 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로, PRIMARY KEY가 컬럼 C 정의에서 지정된 경우, 별도의 절인 PRIMARY KEY(C)절로 지정된 경우와 같은 효과를 봅니다.

테이블이 서브테이블인 경우, 기본 키는 상위 테이블로부터 물려 받게 되므로 기본 키를 지정할 수 없습니다(SQLSTATE 429B3).

다음의 *unique-constraint* 설명에 있는 PRIMARY KEY를 참조하십시오.

UNIQUE

이는 단일 컬럼으로 구성되는 고유 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로, UNIQUE가 컬럼 C 정의에서 지정된 경우, UNIQUE(C)절이 별도의 절로 지정된 경우와 결과가 동일합니다.

테이블이 서브테이블인 경우, 고유 제한조건은 상위 테이블로부터 물려 받게 되므로 고유 제한조건을 지정할 수 없습니다(SQLSTATE 429B3).

다음의 *unique-constraint*의 설명에 있는 UNIQUE를 참조하십시오.

references-clause

이는 단일 컬럼으로 구성되는 외부 키를 정의함에 있어 간편한 방법을 제공합니다. 그러므로, 참조 절이 컬럼 C의 정의에 지정되면, C가 유일하게 식별되는 컬럼인 FOREIGN KEY절의 일부로 참조 절이 지정된 것과 동일한 효력을 갖습니다.

다음의 *referential-constraint* 아래에 있는 *references-clause*을 참조하십시오.

CHECK (*check-condition***)**

이는 단일 컬럼에 적용되는 점검 제한조건을 정의함에 있어 간편한 방법을 제공합니다. 아래에서 CHECK (*check-condition*)을 참조하십시오.

INLINE LENGTH *integer*

이 옵션은 구조화 유형을 사용하여 정의된 컬럼에 대해서만 유효하며

(SQLSTATE 42842) 행에 값의 나머지 부분을 인라인 저장하기 위한 구조화 유형 인스턴스의 최대 바이트 크기를 나타냅니다. 인라인 저장할 수 없는 구조화 유형의 인스턴스는 LOB 값의 처리 방법과 유사하게 기본 테이블 행과는 별도로 저장됩니다. 이 타스크는 자동으로 발생합니다.

구조화 유형 컬럼의 기본 INLINE LENGTH는 해당 유형의 인라인 길이입니다(CREATE TYPE문에 명시적으로 또는 기본값으로 지정됨). 구조화 유형의 INLINE LENGTH가 292보다 작은 경우, 292 값이 컬럼의 INLINE LENGTH로 사용됩니다.

주: 부속 유형의 인라인 길이는 기본 인라인 길이에 계수되지 않으며, 이는 기존의 부속 유형 및 후속 부속 유형을 설명하기 위해 명시적 INLINE LENGTH를 CREATE TABLE 시간에 지정하지 않은 경우에는 부속 유형의 인스턴스가 인라인으로 적합하지 않을 수 있음을 의미합니다.

명시적 INLINE LENGTH 값은 최소한 292이어야 하며 32672를 초과할 수 없습니다(SQLSTATE 54010).

column-default-spec

default-clause

컬럼의 기본값을 지정합니다.

WITH

선택적 키워드

DEFAULT

값이 INSERT에 제공되지 않거나 INSERT 또는 UPDATE의 DEFAULT로 지정된 이벤트에서 기본값을 제공합니다. 기본값이 DEFAULT 키워드 다음에 지정되지 않으면, 기본값은 535 페이지의 표19에 표시된 컬럼의 데이터 유형에 따릅니다.

컬럼이 DATALINK로 정의되어 있는 경우, 기본값을 지정할 수 없습니다(SQLSTATE 42613). 가능한 기본값은 NULL뿐입니다.

컬럼이 입력된 테이블 컬럼에 기초하는 경우, 기본값 정의시 특정 기본값을 지정해야 합니다. 입력된 테이블의 오브젝트 식별자 컬럼에 대한 기본값은 지정할 수 없습니다(SQLSTATE 42997).

컬럼이 구별 유형을 사용하여 정의되면, 컬럼의 기본값은 구별 유형으로 변환되는 원시 데이터 유형의 기본값입니다.

구조화 유형을 사용하여 컬럼을 정의한 경우, *default-clause*를 지정할 수 없습니다(SQLSTATE 42842).

*column-definition*에서 DEFAULT를 생략하면, 컬럼의 기본값으로 널(NULL)값이 사용됩니다. 그러한 컬럼은 NOT NULL로 정의할 경우 유효 기본값을 가지지 않습니다.

default-values

지정될 수 있는 기본값의 특정 유형은 다음과 같습니다.

constant

컬럼의 기본값으로 상수를 지정합니다. 지정된 상수는 다음과 같아야 합니다.

- 제3장에 설명된 지정 규칙에 따라 컬럼에 지정할 수 있는 값을 나타냅니다.
- 컬럼이 부동 소수점 데이터 유형으로 정의되어 있지 않는 한, 부동 소수점 상수가 되지 않습니다.
- 상수가 십진수 상수인 경우 컬럼 데이터 유형의 스케일을 넘어 0이 아닌 자릿수를 갖지 않습니다.(예를 들면, 1. 234는 DECIMAL(5,2) 컬럼에 대한 기본값이 될 수 있습니다.)
- 따옴표, 16진 상수의 X와 같은 도입 문자 및 상수가 *cast-function*의 인수일 때 완전한 함수 이름의 문자 및 괄호를 포함하여 254자 이하로 표시됩니다.

datetime-special-register

컬럼의 기본값으로 INSERT 또는 UPDATE시 날짜시간 특수 레지스터 값(CURRENT DATE, CURRENT TIME

또는 CURRENT TIMESTAMP)을 지정합니다. 컬럼의 데이터 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다(예를 들면, 데이터 유형은 CURRENT DATE 지정시 DATE여야 함).

USER

컬럼의 기본값으로 INSERT 또는 UPDATE시 USER 특수 레지스터 값을 지정합니다. USER가 지정되면, 컬럼의 데이터 유형은 USER의 길이 속성보다 짧지 않은 길이의 문자열이어야 합니다.

NULL

컬럼의 기본 값으로 널(NULL)을 지정합니다. NOT NULL이 지정되었으면, DEFAULT NULL은 동일한 컬럼 정의 내에 지정될 수 있으나, 이로 인해 기본값으로 컬럼을 설정하려고 시도할 때 오류가 발생합니다.

cast-function

이 기본값의 양식은 구별 유형, BLOB 또는 날짜 시간 (DATE, TIME 또는 TIMESTAMP) 데이터 유형으로 정의된 컬럼에만 사용될 수 있습니다. 구별 유형의 경우, BLOB 또는 날짜 시간 유형을 따르는 구별 유형을 제외하고는 함수 이름이 컬럼의 구별 유형 이름과 일치해야 합니다. 스키마 이름으로 규정화된 경우, 이는 구별 유형에 대한 스키마 이름과 같아야 합니다. 규정화되지 않았으면, 함수 해석의 스키마 이름은 구별 유형에 대한 스키마 이름과 같아야 합니다. 기본값이 상수인 날짜 시간 유형을 따르는 구별 유형의 경우, 함수가 사용되어야 하며, 함수 이름에서는 구별 유형의 원시 유형과 SYSIBM의 내재적인 또는 명시적인 스키마 이름이 일치해야 합니다. 다른 날짜시간 컬럼의 경우, 해당 날짜 시간 함수가 사용될 수도 있습니다. BLOB 또는 BLOB에 기초한 구별 유형의 경우, 함수가 사용되어야 하며 함수 이름은 SYSIBM의 내재된 또는 명시적 스키마 이름을 갖는 BLOB이어야 합니다. *cast-function*을 사용하는 예가 552에 나와 있습니다.

constant

인수로서 상수를 지정합니다. 상수는 구별 유형이 아닌 경우, 구별 유형의 원시 유형이나 데이터 유형의 상수에 대한 규칙을 따라야 합니다. *cast-function*가 BLOB인 경우, 상수는 문자열 상수여야 합니다.

datetime-special-register

CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP를 지정합니다. 컬럼의 구별 유형에 대한 원시 유형은 지정된 특수 레지스터에 해당되는 데이터 유형이어야 합니다.

USER

USER 특수 레지스터. 컬럼의 구별 유형에 대한 원시 유형의 데이터 유형은 최소한 8바이트 길이의 문자열 데이터 유형이어야 합니다. *cast-function*가 BLOB인 경우, 길이 속성은 최소한 8바이트여야 합니다.

지정된 값이 유효하지 않으면, 오류(SQLSTATE 42894)가 발생합니다.

GENERATED

DB2가 컬럼의 값을 생성함을 나타냅니다. 컬럼을 생성된 컬럼이나 IDENTITY 컬럼으로 간주하는 경우 GENERATED를 지정해야 합니다.

ALWAYS

DB2가 행이 테이블에 삽입될 때 또는 *generation-expression*의 결과 값이 변할 때마다 항상 컬럼의 값을 생성할 것임을 나타냅니다. 표현식의 결과는 테이블에 저장됩니다. GENERATED ALWAYS는 데이터 전파를 사용 중이 아니거나 로드 해제 및 재로드 조작을 수행 중이 아닌 경우 권장하는 값입니다. GENERATED ALWAYS는 생성된 컬럼의 필수 값입니다.

BY DEFAULT

행이 테이블에 삽입될 때 값을 지정하지 않으면 DB2가 컬럼의 값을 생성할 것임을 나타냅니다. BY DEFAULT는 데이터 전파 사용시 또는 로드 해제/재로드 수행시 권장되는 값입니다.

명시적으로 요구되지는 않지만, 값의 고유성을 보장하기 위해 생성된 컬럼에 대해 고유한 단일 컬럼 색인을 정의해야 합니다.

AS IDENTITY

컬럼이 이 테이블의 식별 컬럼이 되도록 지정합니다.[77] 하나의 테이블은 하나의 IDENTITY 컬럼만을 가질 수 있습니다(SQLSTATE 428C1). IDENTITY 키워드는 해당 컬럼과 연관된 *data-type*이 스케일 0을 갖는 정확한 숫자 유형이거나[78] 소스 유형이 스케일 0을 갖는 정확한 숫자 유형인 사용자 정의 구별 유형인 경우에만 지정할 수 있습니다(SQLSTATE 42815).

식별 컬럼은 내재적으로 NOT NULL입니다.

START WITH *numeric-constant*

식별 컬럼의 첫번째 값을 지정합니다. 이 값은 소수점 오른쪽에 0이 아닌 숫자가 없는한(SQLSTATE 42894) 이 컬럼에 지정할 수 있는 음수 또는 양수 값입니다(SQLSTATE 42820). 기본값은 1입니다.

INCREMENT BY *numeric-constant*

식별 컬럼의 연속 값들 사이 간격을 지정합니다. 이 값은 이 컬럼에 지정할 수 있는 음수 또는 양수 값입니다(SQLSTATE 42820). 이 값은 소수점 오른쪽에 0이 아닌 숫자가 없는 경

77. 여러 파티션이 있는 데이터베이스에서는 식별 컬럼이 지원되지 않습니다(SQLSTATE 42997). 데이터베이스에 대해 둘 이상의 파티션이 있는 경우에는 식별 컬럼을 작성할 수 없습니다. 식별 컬럼을 포함하는 데이터베이스는 둘 이상의 파티션에서 시작할 수 없습니다.

78. 스케일 0을 갖는 SMALLINT, INTEGER, BIGINT 또는 DECIMAL이나 이러한 유형 중 하나에 기초하는 구별 유형은 정확한 숫자 유형으로 고려됩니다. 대조적으로, 단정밀도 또는 배정밀도 부동 소수점은 근사치의 숫자 데이터 유형으로 간주됩니다. 참조 유형은 정확한 숫자로 표시되더라도 식별 유형으로 정의할 수 없습니다.

우(SQLSTATE 42894) 0일 수 없고 큰 정수 상수 값을 초과할 수 없습니다(SQLSTATE 428125).

이 값이 음수인 경우 이 식별 컬럼의 값 순서는 내림차순입니다. 이 값이 양수인 경우 이 식별 컬럼의 값 순서는 오름차순입니다. 기본값은 1입니다.

CACHE 또는 NO CACHE

보다 빠른 액세스를 위해 메모리에 사전 할당된 값 일부를 보존할지 여부를 지정합니다. 식별 컬럼에 새 값이 필요하고 캐쉬 내용을 하나도 사용할 수 없는 경우, 새 캐쉬 블록의 끝을 기록해야 합니다. 그러나 식별 컬럼에 새 값이 필요하고 캐쉬에 사용되지 않는 값이 있다면, 로깅이 필요치 않으므로 해당 식별 값의 할당이 더 빨라집니다. 이것은 성능 조정 옵션입니다.

CACHE *integer-constant*

DB2가 메모리를 사전할당하고 보존하는 식별 순서 값의 양을 지정합니다. 식별 컬럼에 대한 값이 생성될 때 캐쉬에 값을 사전할당하고 저장하면 로깅이 줄어듭니다.

식별 컬럼에 새 값이 필요하고 캐쉬 내용을 하나도 사용할 수 없는 경우, 값 할당에 로그를 기다리는 것도 포함됩니다. 그러나 식별 컬럼에 새 값이 필요하고 캐쉬에 사용되지 않는 값이 있는 경우, 로깅을 수행하지 않고 식별 값을 더 빨리 할당할 수 있습니다.

데이터베이스 비활성화 이벤트시, 정상적으로[79] 또는 시스템 실패로 장애로 인해 확약된 명령문에서 사용되지 않은 캐쉬된 모든 순서 값이 없어집니다. CACHE 옵션에 대해 지정된 값은 데이터베이스 비활성화시 없어질 수 있는 식별 컬럼에 대한 최대 값 수입니다.

79. 데이터베이스가 명시적으로 활성화되지 않았는데(ACTIVATE 명령 또는 API를 사용하여) 마지막 응용프로그램이 데이터베이스로부터 연결해제되면, 내재된 비활성화가 발생합니다.

최소값은 2이고 최대값은 32767입니다(SQLSTATE 42815). 기본값은 CACHE 20입니다.

NO CACHE
식별 컬럼의 값이 사전할당되지 않도록 지정합니다.

이 옵션이 지정되면 식별 컬럼의 값이 캐쉬에 저장되지 않습니다. 이 경우, 새 식별 컬럼 값에 대한 모든 요청이 로깅됩니다.

AS (*generation-expression***)**
컬럼의 정의가 표현식을 근거로 함을 지정합니다.[80] *generation-expression*에는 다음 중 어느 하나도 포함될 수 없습니다 (SQLSTATE 42621):

- 부속 조회
- 컬럼 함수
- 참조 해제 조작 또는 DEREF 함수
- 비 결정적인 사용자 정의 또는 내장 함수
- EXTERNAL ACTION 옵션을 사용하는 사용자 정의 함수
- SCRATCHPAD 옵션을 사용하는 사용자 정의 함수
- READS SQL DATA 옵션을 사용하는 사용자 정의 함수
- 호스트 변수 또는 매개변수 표시문자
- 특수 레지스터
- 컬럼 목록에서 나중에 정의된 컬럼에 대한 참조
- 기타 생성된 컬럼에 대한 참조

컬럼의 데이터 유형은 *generation-expression*의 결과 데이터 유형을 기초로 합니다. CAST 스펙은 특정 데이터 유형을 강요하고 범위(참조 유형의 경우에만)를 제공하는 데 사용할 수 있습니다. *data-type*이 지정된 경우에는 73 페이지의 『제3장 언어 요소』에

80. GENERATED ALWAYS 컬럼에 대한 표현식에 사용자 정의 외부 함수가 포함되는 경우, 이 함수의 실행 가능 파일을 변경하면(제공된 인수에 따라 결과가 변하도록) 데이터 불일치가 발생할 수 있습니다. 이러한 현상은 SET INTEGRITY 문을 사용하여 강제로 새 값을 생성함으로써 막을 수 있습니다.

기술된 지정 규칙에 따라 값이 컬럼에 지정됩니다. 생성된 컬럼은 NOT NULL 컬럼 옵션이 사용되지 않은한 내재적으로 널(NULL) 입력 가능 컬럼으로 고려됩니다. 생성된 컬럼의 데이터 유형은 동등성이 정의된 데이터 유형이어야 합니다. 여기에서 LONG VARCHAR, LONG VARGRAPHIC, LOB 데이터 유형, DATALINK, 구조화 유형 및 이러한 유형에 기초하는 구별 유형의 컬럼은 제외됩니다(SQLSTATE 42962).

OID-column-definition

입력된 테이블에 대한 오브젝트 식별자 컬럼을 정의합니다.

REF IS *OID-column-name* **USER GENERATED**

오브젝트 식별자(OID) 컬럼이 첫번째 컬럼으로서 테이블에 정의되도록 지정합니다. OID는 테이블 계층의 루트 테이블에 필수적입니다(SQLSTATE 428DX). 테이블은 서브테이블이 아닌 입력된 테이블이어야 합니다(OF절이 있어야 합니다)(SQLSTATE 42613). 컬럼 이름이 *OID-column-name*으로 정의되어 있고, 구조화 유형 *type-name1*의 속성 이름과 같을 수 없습니다(SQLSTATE 42711). 컬럼이 유형 REF(*type-name1*), NOT NULL과 함께 정의되어 있고, 시스템에 필요한 고유 색인(기본 색인 이름)이 생성됩니다. 이 컬럼은 *object identifier column* 또는 *OID column*이라고도 합니다. 키워드 USER GENERATED는 행을 삽입할 때 사용자가 OID 컬럼에 대한 초기 값을 제공해야 한다는 것을 나타냅니다. 일단 행이 삽입되면 OID 컬럼은 갱신할 수 없습니다(SQLSTATE 42808).

with-options

입력된 테이블 컬럼에 적용되는 추가 옵션을 정의합니다.

column-name

추가 옵션이 지정될 컬럼의 이름을 지정합니다. *column-name*은 상위 테이블의 컬럼이 아닌 테이블의 컬럼 이름에 해당되어야 합니다(SQLSTATE 428DJ). 컬럼 이름은 명령문에서 하나의 WITH OPTIONS절에만 나타날 수 있습니다(SQLSTATE 42613).

(CREATE TYPE에 있는) 유형 정의의 일부로서 이미 옵션이 지정되어 있는 경우, 여기에 지정된 옵션은 CREATE TYPE에 있는 옵션을 대체합니다.

WITH OPTIONS *column-options*
지정된 컬럼에 대한 옵션을 정의합니다. 앞에서 설명한 *컬럼 옵션*에서 참조하십시오. 테이블이 서브테이블인 경우, 기본 키 또는 고유 제한조건을 지정할 수 없습니다(SQLSTATE 429B3).

DATA CAPTURE
데이터베이스간 데이터 복제에 대한 추가 정보가 로그에 쓰여질지의 여부를 나타냅니다. 이 절은 서브테이블을 작성할 때에는 지정할 수 없습니다(SQLSTATE 42613).

테이블이 유형화된 테이블이면, 이 옵션이 지원되지 않습니다(SQLSTATE 428DH 또는 42HDR).

NONE
추가의 정보가 기록되지 않음을 나타냅니다.

CHANGES
이 테이블에 대한 SQL 변경사항에 따라 추가의 정보가 로그에 기록됨을 나타냅니다. 이 옵션은 이 테이블이 전달되며 캡쳐 프로그램이 로그로부터 이 테이블에 대한 변경사항을 캡쳐하는 데 사용할 경우에 필요합니다.

테이블이 카탈로그 파티션(카탈로그 파티션 이외의 다른 파티션으로 된 다중 파티션 노드 그룹 또는 노드 그룹) 이외의 다른 파티션에서 데이터를 허용하도록 정의되면, 이 옵션은 지원되지 않습니다(SQLSTATE 42997).

테이블의 스키마 이름(내재된 또는 명시적)이 18바이트보다 큰 경우에는 이 옵션이 지원되지 않습니다(SQLSTATE 42997).

복사 사용에 대한 정보는 *관리 안내서*와 *복제 안내 및 참조서*에서 찾을 수 있습니다.

IN *tablespace-name1*
테이블이 작성되는 테이블 공간을 식별합니다. 테이블 공간이 있어야

하고 이 테이블 공간은 명령문의 권한 부여 ID가 USE 특권을 가지는 REGULAR 테이블 공간이어야 합니다. 어떤 다른 테이블 공간도 지정되지 않으면, 모든 테이블 부분이 이 테이블 공간에 저장됩니다. 테이블 공간이 테이블 계층의 루트 테이블로부터 물려받은 것이므로, 이 절은 서브테이블을 작성할 때에는 지정할 수 없습니다(SQLSTATE 42613). 이 절이 지정되지 않으면, 테이블에 대한 테이블 공간은 다음과 같이 결정됩니다.

```
IF table space IBMDEFAULTGROUP over which the user has USE privilege
  exists with sufficient page size
    THEN choose it
 ELSE IF a table space over which the user has USE privilege
  exists with sufficient page size
  (see below when multiple table spaces qualify)
    THEN choose it
 ELSE issue an error (SQLSTATE 42727).
```

둘 이상의 테이블 공간이 ELSE IF 조건으로 식별되는 경우, 해당 명령문의 권한 부여 ID가 USE 특권을 갖으며 가장 작은 *크기*의 충분한 페이지가 있는 테이블 공간을 선택하십시오. 둘 이상의 테이블 공간이 규정되었다면 USE 특권이 부여된 사용자에 따라 환경설정이 제공됩니다.

1. 권한 부여 ID
2. 권한 부여 ID가 포함된 그룹
3. PUBLIC

여전히 둘 이상의 테이블 공간이 규정되었다면 데이터베이스 관리 프로그램에서 최종 선택을 합니다.

테이블 공간 결정은 다음의 경우에 변경할 수 있습니다.

- 테이블 공간이 삭제 또는 작성되었을 때
- USE 특권이 권한 부여 또는 권한 취소되었을 때

테이블의 페이지 *크기*가 충분한지 여부는 행의 바이트 합계 또는 컬럼 수에 의해 결정됩니다. 853 페이지의 『행 *크기*』에서 자세한 설명을 참조하십시오.

tablespace-options:

색인이나 긴 컬럼 값이 저장되는 테이블 공간을 지정합니다. 테이블의 유형에 대해 861 페이지의 『CREATE TABLESPACE』에서 자세한 내용을 참조하십시오.

INDEX IN *tablespace-name2*

테이블의 색인이 작성될 테이블 공간을 식별합니다. 이 옵션은 IN 절에 지정된 1차 테이블 공간이 DMS 테이블 공간일 경우에만 허용됩니다. 지정된 테이블 공간이 있어야 하고 이 테이블 공간은 명령문의 권한 부여 ID가 USE 특권을 가지는 일반 DMS 테이블 공간이어야 하며 *tablespace-name1*과 같은 노드 그룹에 있어야 합니다(SQLSTATE 42501).

테이블 색인이 들어 갈 테이블 공간을 지정하는 것은 테이블 작성시에만 수행할 수 있습니다. 색인용 테이블 공간에 대한 USE 특권 검사는 테이블 작성시에만 수행됩니다. 데이터베이스 관리 프로그램의 경우 나중에 색인 작성될 때 CREATE INDEX 문의 권한 부여 ID에게 테이블 공간에 대한 USE 특권이 필요 없습니다.

LONG IN *tablespace-name3*

긴 컬럼의 값(LONG VARCHAR, LONG VARGRAPHIC, LOB 데이터 유형, 소스 유형으로 이 중 하나를 가지는 구별 유형이나 인라인으로 저장할 수 없는 값을 가지는 사용자 정의 구조화 유형으로 정의된 컬럼)이 저장될 테이블 공간을 식별합니다. 이 옵션은 IN절에 지정된 1차 테이블 공간이 DMS 테이블 공간일 경우에만 허용됩니다. 테이블 공간이 있어야 하고 이 테이블 공간은 명령문의 권한 부여 ID가 USE 특권을 가지는 LONG DMS 테이블 공간이어야 하며 *tablspace-name1*의 같은 노드 그룹에 있어야 합니다(SQLSTATE 42838).

테이블의 LONG 및 LOB 컬럼이 포함될 테이블 공간을 지정하는 작업은 테이블이 작성될 때에만 수행할 수 있음에 유의하십시오. LONG 및 LOB 컬럼의 테이블 공간에 대한 USE 특권 검사는 테이블 작성시에만 수행됩니다. 데이터베이스 관리 프로그램의

경우, 나중에 Long 또는 LOB가 추가될 때 ALTER TABLE 문의 권한 부여 ID에게 테이블 공간에 대한 USE 특권이 필요 없습니다.

PARTITIONING KEY (*column-name*,...)

테이블의 데이터가 파티션된 경우에 사용되는 파티션 키를 지정합니다. 각 *column-name*은 테이블의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별되어서는 안됩니다. LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK와 같은 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형을 갖는 컬럼은 파티션 키의 일부로 사용할 수 없습니다(SQLSTATE 42962). 파티션 키가 테이블 계층의 루트 테이블로부터 물려받은 것이므로, 서브테이블인 테이블에 대해 파티션 키를 지정할 수 없습니다(SQLSTATE 42613).

이 절이 지정되지 않고 이 테이블이 여러 파티션 노드 그룹에 상주할 경우, 파티션 키는 다음과 같이 정의됩니다.

- 테이블이 입력된 테이블인 경우, 오브젝트 식별자 컬럼입니다
- 기본 키가 지정되는 경우, 기본 키의 첫번째 컬럼은 파티션 키입니다
- 그렇지 않을 경우, LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK와 같은 이러한 유형에 기초하는 구별 유형이 아닌 첫번째 컬럼은 파티션 키입니다.

기본 파티션 키 요건을 충족시키는 컬럼이 없을 경우, 파티션 키 없이 테이블이 작성됩니다. 이러한 테이블은 단일 파티션 노드 그룹에 정의된 테이블 공간에서만 가능합니다.

단일 파티션 노드 그룹에 정의된 테이블 공간에 있는 테이블의 경우, LONG이 아닌 유형의 컬럼을 임의로 콜렉션하여 사용하여 파티션 키를 정의할 수 있습니다. 이 매개변수를 지정하지 않을 경우, 파티션 키가 작성되지 않습니다.

파티션 키와 관련된 제한사항에 대해서는 847 페이지의『규칙』에서 참조하십시오.

USING HASHING

데이터 분배를 위한 파티션 방법으로 해싱 함수의 사용을 지정합니다. 이것이 유일하게 지원되는 파티션 방법입니다.

REPLICATED

테이블에 저장된 데이터가 테이블이 정의된 테이블 공간의 노드그룹에 있는 각 데이터베이스 파티션에 물리적으로 복제됨을 지정합니다. 이는 테이블 내의 모든 데이터의 사본이 각각의 데이터베이스 파티션에 존재함을 의미합니다. 이 옵션은 요약 테이블에 대해서만 지정할 수 있습니다(SQLSTATE 42997).

NOT LOGGED INITIALLY

테이블이 작성된 동일한 작업 단위(UOW)에서 삽입, 삭제, 갱신, 색인 작성, 색인 삭제 또는 테이블 교체 연산을 통해 테이블에 가한 변경은 로그되지 않습니다. 이 옵션 사용시의 기타 고려사항에 대해서는 849 페이지의 『주』에서 참조하십시오.

모든 카탈로그 변경사항 및 저장영역 관련 정보는 후속되는 단위 작업에서 테이블에 행해지는 모든 연산과 마찬가지로 로그됩니다.

NOT LOGGED INITIALLY 속성으로 상위 테이블을 참조하는 테이블에 대해서는 외부 키 제한조건을 정의할 수 없습니다. 이 절은 서브테이블을 작성할 때에는 지정할 수 없습니다(SQLSTATE 42613).

주: 세이브포인트 요청으로의 구간 복원은 NOT LOGGED INITIALLY 테이블 작성과 동일한 작업 단위(UOW)에서 발생할 수 없습니다. 그럴 경우, 오류가 발생하고(SQLSTATE 40506) 작업 단위 전체가 구간 복원됩니다.

unique-constraint

고유성 제한조건이나 기본 키 제한조건을 정의합니다. 테이블이 파티션 키를 가질 경우 이 때 고유 키 또는 기본 키는 파티션 키의 슈퍼 세트이어야 합니다. 서브테이블인 테이블에 대해 고유 키 또는 기본 키 제한조건을 지정할 수 없습니다(SQLSTATE 429B3). 테이블이 루트 테이블인 경우, 제한조건은 테이블 및 모든 서브테이블에 적용됩니다.

CONSTRAINT *constraint-name*

기본 키 또는 고유성 제한조건을 명명합니다. 826 페이지를 참조하십시오.

UNIQUE (*column-name*,...)

식별된 컬럼들로 구성되는 고유 키를 정의합니다. 식별된 컬럼은 NOT NULL로 정의되어야 합니다. 각 *column-name*은 테이블의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별되어서는 안 됩니다.

식별된 컬럼의 수는 16을 초과할 수 없으며 저장된 길이의 합은 1024를 초과할 수 없습니다(저장 길이에 대해서는 853 페이지의 『바이트 계수』 참조). 개별적인 컬럼의 길이는 255바이트를 초과해서는 안됩니다. 이 길이는 데이터 전용이고 널(NULL) 바이트의 영향을 받지 않습니다. 컬럼의 최대 데이터 길이는 컬럼의 널(NULL) 입력 가능 여부에 관계없이 255바이트입니다. 컬럼의 길이 속성이 255바이트 한계내에 들어갈만큼 작은 경우에도 LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK와 같은 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형은 고유 키의 일부로 사용할 수 없습니다(SQLSTATE 42962).

고유 키의 컬럼 집합은 기본 키 또는 다른 고유 키의 컬럼 집합과 동일할 수 없습니다(SQLSTATE 01543). [81]

테이블이 서브테이블인 경우, 고유 제한조건은 상위 테이블로부터 물려 받게 되므로 고유 제한조건을 지정할 수 없습니다(SQLSTATE 429B3).

카탈로그에 기록된 대로 테이블의 설명에는 고유 키와 이의 고유 색인이 포함됩니다. 고유 색인은 컬럼에 대해 자동으로 작성됩니다(각 컬럼에 대해 오름차순으로). 색인의 이름은 테이블이 작성된 스키마의 기존 색인과 상충하지 않는 경우, *constraint-name*과 동일합니다. 색인 이름이 상충하는 경우, 이름은 문자 시각이 후속되는 SQL(*yymmddhhmmssxxx*)이 되며, SYSIBM을 스키마 이름으로 갖습니다.

PRIMARY KEY (*column-name*,...)

식별된 컬럼들로 구성되는 기본 키를 정의합니다. 절은 두 번 이상 지정되

81. LANGLEVEL이 SQL92E 또는 MIA인 경우에는 오류가 리턴됩니다(SQLSTATE 42891).

어서는 안되며, 식별된 컬럼은 NOT NULL로 정의되어야 합니다. 각 *column-name*은 테이블의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별되어서는 안 됩니다.

식별된 컬럼의 수는 16을 초과할 수 없으며 저장된 길이의 합은 1024를 초과할 수 없습니다(저장 길이에 대해서는 853 페이지의 『바이트 계수』 참조). 개별적인 컬럼의 길이는 255바이트를 초과해서는 안됩니다. 이 길이는 데이터 전용이고 널(NULL) 바이트의 영향을 받지 않습니다. 컬럼의 최대 데이터 길이는 컬럼의 널(NULL) 입력 가능 여부에 관계없이 255바이트입니다. 컬럼의 길이 속성이 255바이트 한계내에 들어갈 만큼 작은 경우에도 LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK와 같은 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형은 기본 키의 일부로 사용할 수 없습니다(SQLSTATE 42962).

기본 키의 컬럼 세트는 고유 키의 컬럼 세트와 같을 수 없습니다(SQLSTATE 01543). [81]

하나의 테이블에서 단 하나의 기본 키만 정의할 수 있습니다.

테이블이 서브테이블인 경우, 기본 키는 상위 테이블로부터 물려 받게 되므로 기본 키를 지정할 수 없습니다(SQLSTATE 429B3).

카탈로그에 기록된 대로 테이블의 설명에는 기본 키와 해당되는 1차 색인이 포함됩니다. 고유 색인은 컬럼에 대해 자동으로 작성됩니다(각 컬럼에 대해 오름차순으로). 색인의 이름은 테이블이 작성된 스키마의 기존 색인과 상충하지 않는 경우, *constraint-name*과 동일합니다. 색인 이름이 상충하는 경우, 이름은 문자 시각이 후속되는 SQL(*yymmddhhmmssxxx*)이 되며, SYSIBM을 스키마 이름으로 갖습니다.

테이블에 파티션 키가 있는 경우 *unique-constraint*의 컬럼은 파티션 키 컬럼의 상위 집합이어야 합니다. 컬럼 순서는 중요하지 않습니다.

referential-constraint

참조 제한조건을 정의합니다.

CONSTRAINT *constraint-name*

참조 제한조건을 명명합니다. 826 페이지를 참조하십시오.

FOREIGN KEY (*column-name*,...)

지정된 *constraint-name*을 가진 참조 제한조건을 정의합니다.

T1은 명령문의 오브젝트 테이블을 나타낸다고 합시다. 참조 제한조건의 외부 키는 식별되는 컬럼으로 구성됩니다. 컬럼 이름 목록의 각 이름은 T1의 컬럼을 식별하고, 같은 컬럼은 두 번 이상 식별되어서는 안 됩니다. 식별된 컬럼의 수는 16을 초과할 수 없으며 저장된 길이의 합은 1024를초과할 수 없습니다(저장 길이에 대해서는 853 페이지의『바이트 계수』참조). LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK와 같은 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형은 외부 키의 일부로 사용할 수 없습니다(SQLSTATE 42962). 상위 키에 있는 것과 동일한 수의 외부 키 컬럼이 있어야 하며, 해당 컬럼의 데이터 유형은 호환 가능해야 합니다(SQLSTATE 42830). 데이터 유형이 호환 가능할 경우 두 컬럼 설명이 호환 가능합니다(두 컬럼이 숫자, 문자열, 그래픽, 날짜/시간 또는 동일한 구별 유형).

references-clause

참조 제한조건에 대한 상위 테이블과 상위 키를 지정합니다.

REFERENCES *table-name*

REFERENCES절에 지정된 테이블은 카탈로그에 설명된 기본 테이블을 식별해야 하나, 카탈로그 테이블은 식별하지 말아야 합니다.

참조 제한조건의 외부 키, 상위 키, 상위 테이블이 이전에 지정된 참조 제한조건의 외부 키, 상위 키, 상위 테이블과 동일할 경우, 참조 제한조건이 중복됩니다. 중복 참조 제한조건은 무시되며 경고가 표시됩니다(SQLSTATE 01543).

다음 설명에서, T2는 식별된 상위 테이블을 나타내고, T1은 작성중[82]인 테이블을 나타냅니다.(T1 및 T2가 동일한 테이블일 수도 있습니다.)

지정되는 외부 키는 T2의 상위 키와 컬럼 수가 같아야 하며, 외부 키의 *n*번째 컬럼의 설명은 상위 키의 *n*번째 컬럼의 설명과 호환해야 합니다. 날짜시간 컬럼은 이 규칙에 대해 문자열 컬럼과 호환성이 있는 것으로 간주되지 않습니다.

82. 또는 변경중(이 절이 ALTER TABLE문의 설명에서 참조된 경우)

(*column-name*,...)

참조 제한조건의 상위 키는 식별되는 컬럼으로 구성됩니다. 각 *column-name*은 T2의 컬럼을 식별하는 규정화되지 않은 이름이어야 합니다. 같은 컬럼은 한 번 이상 식별될 수 없습니다.

컬럼 이름의 목록은 T2에 존재하는 기본 키나 고유 제한조건의 컬럼 세트(순서는 관계없음)와 일치해야 합니다(SQLSTATE 42890). 컬럼 이름 목록이 지정되지 않을 경우, T2에 기본 키가 있어야 합니다(SQLSTATE 42888). 컬럼 이름 목록을 누락하면 원래 지정된 순서로 그 기본 키의 컬럼 스펙이 내재됩니다.

FOREIGN KEY절에 의해 지정된 참조 제한조건은 T2가 상위이고 T1이 종속인 관계를 정의합니다.

rule-clause

종속 테이블에서 취할 조치를 지정합니다.

ON DELETE

상위 테이블의 행이 삭제될 때 종속 테이블에서 취할 조치를 지정합니다. 네 가지의 가능한 조치가 있습니다.

- NO ACTION(기본값)
- RESTRICT
- CASCADE
- SET NULL

삭제 규칙은 T2의 행이 DELETE의 오브젝트이거나 삭제 작업이 전달되고 그 행에 T1의 종속사항이 있을 때 적용됩니다. *p*는 T2의 그러한 행을 나타낸다고 합시다.

- RESTRICT나 NO ACTION을 지정한 경우, 오류가 발생하여 어떤 행도 삭제되지 않습니다.
- CASCADE를 지정한 경우, 삭제 작업은 T1에 있는 *p*의 종속사항으로 전달됩니다.
- SET NULL이 지정되면, T1에 있는 *p*의 각 종속사항에 대한 외부 키의 널(NULL) 값 입력 가능한 각 컬럼은 널(NULL) 로 설정됩니다.

외부 키의 일부 컬럼이 널(NULL) 값을 허용하지 않으면 SET NULL을 지정해서는 안 됩니다. 절을 생략하는 것은 ON DELETE NO ACTION의 지정을 암시하는 것입니다.

두 개 이상의 테이블과 관련되는 순환은 순환에서의 모든 삭제 규칙이 CASCADE가 아닐 경우 테이블이 그 자체에 대해 삭제 연결되지 않도록 해야 합니다. 그러므로, 새로운 관계가 순환을 형성할 수 있고 T2가 이미 T1에 대해 삭제 연결되어 있는 경우, CASCADE 삭제 규칙을 갖고 있고 순환의 모든 다른 삭제 규칙이 CASCADE인 경우에만 제한조건을 정의할 수 있습니다.

T1이 복수의 경로를 통해 T2에 대해 삭제 연결되는 경우, T1이 종속이고 그 경로에서 일부 또는 모두를 형성하는 관계는 같은 삭제 규칙을 갖고 있어야 하고, SET NULL이 아니어야 합니다. NO ACTION과 RESTRICT 조치는 동일하게 처리됩니다. 따라서 T1이 *r*의 삭제 규칙 관계에서 T3에 종속되는 경우 다음과 같은 조건하에서 *r*이 SET NULL이라면 참조 제한조건을 정의할 수 없습니다.

- T2와 T3가 같은 테이블입니다.
- T2는 T3의 종속이고 T3로부터의 행의 삭제가 T2로 연쇄됩니다.
- T3가 T2의 종속이고 T2로부터의 행의 삭제가 T3로 연쇄됩니다.
- T2와 T3 둘다가 같은 테이블의 종속이고, 그 테이블로부터의 행 삭제가 T2 및 T3로 연쇄됩니다.

*r*이 SET NULL이외의 다른 규칙이면, 참조 제한조건을 정의할 수는 있으나, FOREIGN KEY절에 암시적이나 명시적으로 지정되는 삭제 규칙은 *r*과 같아야 합니다.

위의 규칙을 참조 제한조건에 적용하는 경우 상위 테이블이나 종속 테이블이 유형화 테이블 계층의 구성원이라면, 각 계층의 테이블에 적용되는 모든 참조 제한조건이 고려됩니다.

ON UPDATE

상위 테이블의 행이 갱신될 때 종속 테이블에서 취할 조치를 지정합니다. 이 절은 생략 가능합니다. ON UPDATE NO ACTION은 기본값이며, ON UPDATE RESTRICT는 유일한 대체값입니다.

NO ACTION과 RESTRICT의 차이점은 849 페이지의 『주』의 CREATE TABLE 아래에서 설명합니다.

check-constraint

점검 제한조건을 정의합니다. *check-constraint*은 거짓이 아닌 지를 평가해야 하는 *search-condition*입니다.

CONSTRAINT *constraint-name*

점검 제한조건을 명명합니다. 826 페이지를 참조하십시오.

CHECK (*check-condition*)

점검 제한조건을 정의합니다. *check-condition*은 다음을 제외한 *search-condition*입니다.

- 컬럼 참조는 작성되는 테이블의 컬럼이어야 합니다.
- *search-condition*에는 TYPE 술어가 포함될 수 없습니다.
- 다음 중 어느 것도 포함될 수 없습니다(SQLSTATE 42621).
  - 부속 조회
  - 범위 지정된 참조 인수가 오브젝트 식별자(OID) 컬럼이 아닌 참조 해제 조작이나 DEREF 함수.
  - SCOPE 절이 있는 CAST 권장 스펙
  - 컬럼 함수
  - 결정할 수 없는 함수
  - 외부 조치를 가지도록 정의된 함수
  - SCRATCHPAD 옵션을 사용하는 사용자 정의 함수
  - READS SQL DATA 옵션을 사용하는 사용자 정의 함수
  - 호스트 변수
  - 매개변수 표시문자
  - 특수 레지스터

- 별명
- 식별 컬럼이 아닌 다른 생성된 컬럼에 대한 참조

*column-definition*의 부분으로 점검 제한조건이 지정된 경우, 컬럼 참조는 같은 컬럼에 대해서만 이루어질 수 있습니다. 테이블 정의의 부분으로 지정된 점검 제한조건은 CREATE TABLE문에 이전에 정의한 컬럼을 식별하는 컬럼 참조를 가질 수 있습니다. 비일관성, 중복 조건 또는 동등한 조건에 대해 점검 제한조건이 점검되지 않습니다. 그러므로, 모순되거나 불필요한 점검 제한조건이 정의되면 실행시 오류가 발생할 수 있습니다.

점검 조건 "IS NOT NULL"이 지정될 수 있으나, 컬럼의 NOT NULL 속성을 사용하여 직접 널(NULL) 값 입력 가능성이 강제되는 것이 좋습니다. 예를 들어, 급여가 널(NULL)로 설정된 경우에는 CHECK 제한조건을 만족시켜야 하거나 알 수 없어야 하며 이 때 급여를 열 수 없기 때문에, CHECK (salary + bonus > 30000)이 사용됩니다. 그러나, 급여가 NULL로 설정된 경우 CHECK (salary IS NOT NULL)는 거짓이고 제한조건에 위반되는 것으로 처리됩니다.

점검 제한조건은 테이블의 행이 삽입되거나 갱신될 때 시행됩니다. 테이블에 지정된 점검 제한조건은 그 테이블의 모든 서브테이블에 자동으로 적용됩니다.

규칙

- 모든 구조화 유형 컬럼의 인라인 길이를 비롯한 컬럼의 바이트 계수 합은 테이블 공간의 페이지 크기에 기초하는 행 크기 한계보다 커서는 안됩니다 (SQLSTATE 54010). 더 자세히 알려면 853 페이지의 『바이트 계수』 및 1255 페이지의 표33에서 참조하십시오. 입력된 테이블인 경우, 바이트 합계는 테이블 계층의 루트 테이블 컬럼과 테이블 계층에 있는 모든 서브테이블에 의해 도입된 모든 추가 컬럼에 적용됩니다. (널(NULL) 입력이 불가능한 것으로 정의된 경우에도, 바이트 계산상 추가 서브테이블 컬럼을 널(NULL) 입력이 가능한 것으로 간주해야 합니다.) 각 행이 속해 있는 서브테이블을 찾기 위해 4바이트의 추가 오버헤드도 있습니다.

- 테이블의 컬럼 수는 1 012를 초과할 수 없습니다(SQLSTATE 54011). 입력된 테이블의 경우, 테이블 계층에 있는 모든 서브테이블의 총 유형 속성 수는 1010을 초과할 수 없습니다.
- 입력된 테이블의 오브젝트 식별자 컬럼은 갱신할 수 없습니다(SQLSTATE 42808).
- 테이블의 파티션 키 컬럼은 갱신할 수 없습니다(SQLSTATE 42997).
- 테이블에 정의된 고유 키 또는 기본 키 제한조건은 파티션 키의 슈퍼 세트여야 합니다(SQLSTATE 42997).
- 파티션 키의 널(NULL) 값 입력 가능 컬럼은 ON DELETE SET NULL을 사용하여 관계가 정의되면 외부 키 컬럼으로 포함시킬 수 없습니다(SQLSTATE 42997).
- 다음 테이블은 *파일 링크 옵션*에서 지원되는 DATALINK 옵션의 조합을 제공합니다(SQLSTATE 42613).

표 22. 유효한 DATALINK 파일 제어 옵션 조합

INTEGRITY	READ PERMISSION	WRITE PERMISSION	RECOVERY	ON UNLINK
ALL	FS	FS	NO	적용 불가능
ALL	FS	BLOCKED	NO	RESTORE
ALL	FS	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	NO	RESTORE
ALL	DB	BLOCKED	NO	DELETE
ALL	DB	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	YES	DELETE

다음 규칙은 파티션된 데이터베이스에만 적용됩니다.

- LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK와 같은 이러한 유형 중 하나에 기초하는 구별 유형 또는 구조화 유형을 가지는 컬럼으로만 구성된 테이블은 단일 파티션 노드 그룹에 정의된 테이블 공간에만 작성할 수 있습니다.
- 다중 파티션 노드 그룹에 정의된 테이블 공간에 있는 테이블의 파티션 키 정의는 변경할 수 없습니다.
- 입력된 테이블의 파티션 키 컬럼은 OID 컬럼이어야 합니다.

주

- 아직 존재하지 않는 스키마 이름을 사용하여 색인을 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 외부 키가 지정되는 경우:
  - 상위 테이블에서 삭제 사용을 갖고 있는 모든 패키지가 유효하지 않게 됩니다.
  - 상위 키내에서 적어도 한 컬럼에 대해 갱신 사용된 적이 있는 모든 패키지는 무효화됩니다.
- 서브테이블을 작성하면 테이블 계층상 다른 테이블에 종속된 패키지는 모두 무효화됩니다.
- 각기 4 000과 2 000 보다 큰 VARCHAR 및 VARGRAPHIC 컬럼은 SYSFUN 스키마에 있는 함수의 입력 매개변수로 사용되어서는 안 됩니다. 이들 길이를 초과하는 인수 값으로 함수가 호출될 때에는 오류가 발생할 것입니다(SQLSTATE 22001).
- 참조 제한조건에 대한 삭제 또는 갱신 규칙으로 NO ACTION 또는 RESTRICT를 사용하는 것은 제한조건이 실행될 때를 판별합니다. RESTRICT의 삭제 또는 갱신 규칙은 CASCADE 또는 SET NULL과 같은 수정 규칙과 함께 이러한 참조 제한조건을 포함하여 다른 모든 제한조건 이전에 실행됩니다. NO ACTION의 삭제 또는 갱신 규칙은 다른 참조 제한조건 후에 실행됩니다. 삭제 또는 갱신중에 다른 것을 수행할 수 있는 경우는 매우 드뭅니다. 다른 특징이 명백해지는 한 예로, 관련 테이블의 UNION ALL로 정의된 뷰에서 행을 DELETE하는 것을 들 수 있습니다.

```
Table T1 is a parent of table T3, delete rule as noted below
Table T2 is a parent of table T3, delete rule CASCADE
CREATE VIEW V1 AS SELECT * FROM T1 UNION ALL SELECT * FROM T2
DELETE FROM V1
```

  테이블 T1이 삭제 규칙 RESTRICT를 갖는 테이블 T3의 상위 테이블인 경우, T3에 T1의 상위 키를 갖는 하위 행이 하나라도 있으면 RESTRICT 위반이 발생합니다(SQLSTATE 23001).

테이블 T1이 테이블 T3의 상위 테이블이고, 삭제 규칙은 NO ACTION인 경우, NO ACTION 삭제 규칙이 T1으로부터의 삭제를 위해 실행되기 전에 T2로부터 행을 삭제할 때 CASCADE의 삭제 규칙이 하위를 삭제합니다. T2로부터의 삭제가 T3내 T1의 상위 키에 대한 모든 하위 행을 삭제하지 않는 경우, 제한조건 위반이 발생합니다(SQLSTATE 23504).

리턴되는 SQLSTATE는 삭제 또는 갱신 규칙이 RESTRICT 또는 NO ACTION인지에 따라 다름을 주의하십시오.

- 다중 파티션 노드 그룹에 정의된 테이블 공간에 있는 테이블의 경우, 파티션 키를 선택할 때 테이블의 배치를 고려해야 합니다. 다음은 고려해야 할 항목의 목록입니다.
  - 배치하려면 테이블이 동일한 노드 그룹에 있어야 합니다. 테이블 공간은 다를 수 있지만, 동일한 노드 그룹에 정의되어야 합니다.
  - 테이블의 파티션 키는 동일한 수의 컬럼을 가져야 하며 해당 키 컬럼은 배치와 호환되는 파티션이어야 합니다. 132 페이지의 『파티션 호환성』에서 자세한 정보를 참조하십시오.
  - 파티션 키의 선택은 조인의 성능에도 영향을 미칩니다. 테이블이 다른 테이블과 자주 조인되는 경우, 조인 컬럼을 두 테이블에 대한 파티션 키로 간주해야 합니다.
- FILE LINK CONTROL 속성을 가진 DATALINK 컬럼이 테이블에 존재할 때에는 NOT LOGGED INITIALLY절이 사용될 수 없습니다(SQLSTATE 42613).
- NOT LOGGED INITIALLY 옵션은 대체 소스(다른 테이블이나 파일)의 데이터를 통해 대량의 결과 집합을 작성해야 하는 상황에서 유용하며 테이블의 복구는 필요하지 않습니다. 이 옵션을 사용하면 데이터 로깅에 따른 오버헤드가 절약됩니다. 다음 고려사항은 이 옵션이 지정될 때 적용됩니다.
  - 작업 단위(UOW)가 확약될 때, 그 작업 단위 중에 테이블에 대해 수행된 모든 변경은 디스크로 이동됩니다.
  - 롤 포워드(Rollforward) 유틸리티를 수행하는 중에 데이터베이스의 테이블이 로드 유틸리티를 통해 채워지거나 NOT LOGGED INITIALLY 옵션을 통해 작성되었음을 나타내는 로그 레코드가 표시되는 경우, 테이블은 사용 불가능으로 표시됩니다. 차후 DROP TABLE 로그를 만나게 되면 Rollforward

유틸리티는 그 테이블을 삭제합니다. 그렇지 않으면, 데이터베이스가 복구된 후, 테이블에 액세스하려고 하면 오류가 표시됩니다(SQLSTATE 55019). 테이블 삭제만 허용됩니다.

– 일단 이러한 테이블이 데이터베이스나 테이블 공간 백업의 일부로 백업되면, 테이블을 복구할 수 있습니다.

- ENABLE QUERY OPTIMIZATION으로 정의된 REFRESH DEFERRED 요약 테이블은 CURRENT REFRESH AGE가 ANY로 설정된 경우 조회 처리를 최적화하는 데 사용할 수 있습니다. 최적화를 위해서 항상 ENABLE QUERY OPTIMIZATION으로 정의된 REFRESH IMMEDIATE 요약 테이블이 고려됩니다. 최적화를 위해 REFRESH DEFERRED 또는 REFRESH IMMEDIATE 요약 테이블을 사용할 수 있도록 하려면, fullselect가 이미 설명된 규칙 외에도 특정 규칙을 준수해야 합니다. 이 fullselect는 다음과 같아야 합니다.
  – GROUP BY절을 가진 부속 선택이거나 단일 테이블 참조를 가진 부속 선택이어야 합니다.
  – 선택 목록 어디에도 DISTINCT가 포함되면 안 됩니다
  – 특수 레지스터를 포함해서는 안 됩니다
  – 결정할 수 없는 함수를 포함해서는 안 됩니다

  요약 테이블 작성시, 지정된 조회가 이러한 규칙을 지키지 않으면 경고가 리턴됩니다(SQLSTATE 01633).
- 요약 테이블이 REFRESH IMMEDIATE로 정의되면, 기초가 되는 테이블을 삽입, 갱신 또는 삭제한 결과 생긴 변경을 적용하려고 시도할 때에 오류가 발생할 수도 있습니다. 오류는 기초가 되는 테이블의 삽입, 갱신 또는 삭제가 실패하게 할 것입니다.
- 참조 제한조건은 상위 테이블이나 종속 테이블이 상위 계층의 일부가 되는 방법으로 정의할 수 있습니다. 이러한 경우, 참조 제한조건의 효과는 다음과 같습니다.

  1. INSERT, UPDATE 및 DELETE문의 영향:
     – 참조 제한조건이 있는 경우 PT가 상위 테이블이고 DT가 종속 테이블이라면, 이 제한조건은 널(NULL)이 아닌 외부 키를 갖는 DT(또는 그의 하위 테이블)의 각 행에 대해 일치하는 상위 키를 갖는 행이 PT에 있는

지를 확인합니다. 이 규칙은 조치 시작 방법에 관계없이 PT 또는 DT의 행에 영향을 주는 어느 조치에 대해서도 강요됩니다.

2. DROP TABLE문의 영향:
   - 삭제된 테이블이 상위 테이블이거나 종속 테이블인 참조 제한조건의 경우에는 해당 제한조건이 삭제됩니다.
   - 삭제된 테이블의 수퍼 테이블이 상위 테이블인 참조 제한조건의 경우에는 삭제된 테이블의 행을 수퍼 테이블에서 삭제할 것을 고려합니다. 참조 제한조건이 검사되고 삭제된 행 각각에 대해 삭제 규칙이 호출됩니다.
   - 삭제된 테이블의 수퍼 테이블이 종속 테이블인 참조 제한조건의 경우에는 해당 제한조건이 검사되지 않습니다. 종속 테이블에서 행을 삭제하면 참조 제한조건의 위반이 일어날 수 없습니다.

- ***실행 불능 요약 테이블:*** 실행 불능 요약 테이블은 더 이상 SQL문에 사용할 수 없는 테이블입니다. 다음의 경우에 요약 테이블은 실행 불능 상태가 됩니다.
  - 요약 테이블 정의가 종속된 특권이 권한 취소됩니다.
  - 요약 테이블 정의가 종속된 오브젝트(예: 테이블, 별명 또는 함수)가 삭제됩니다.

  실제 용어에서, 실행 불능 요약 테이블은 요약 테이블 정의가 우연히 삭제된 뷰입니다. 예를 들어, 별명이 삭제되면 그 별명을 사용하여 정의된 요약 테이블은 실행 불능 상태가 됩니다. 요약 테이블에 종속된 모든 패키지는 더 이상 유효하지 않습니다.

  실행 불능 요약 테이블이 명시적으로 재작성되거나 삭제될 때까지, 그 실행 불능 요약 테이블을 사용하는 명령문은 컴파일될 수 없으나 CREATE ALIAS, CREATE TABLE, DROP TABLE 및 COMMENT ON TABLE문은 예외입니다(SQLSTATE 51024). 실행 불능 요약 테이블이 명시적으로 삭제될 때까지, 다른 뷰, 기본 테이블 또는 별명을 작성하는 데 규정화된 이름을 사용할 수 없습니다(SQLSTATE 42710).

  작동 불능 요약 테이블은 실행 불능 요약 테이블의 정의 텍스트를 사용하여 CREATE TABLE문을 발행함으로써 재작성될 수 있습니다. 이 요약 테이블 조회 텍스트는 SYSCAT.VIEWS 카탈로그의 TEXT 컬럼에 저장됩니다. 실행 불능 요약 테이블을 재작성하면 다른 사용자에게 필요한 특권을 명시적으로 권한

부여해야 하는 데, 이는 요약 테이블이 실행 불능으로 표시되면 요약 테이블에 있는 모든 권한 부여 레코드가 삭제되기 때문입니다. 실행 불능 요약 테이블을 재작성하기 위해 이를 명시적으로 삭제할 필요는 없음을 주의하십시오. 실행 불능 요약 테이블과 동일한 *테이블 이름*을 가진 요약 테이블을 정의하는 CREATE TABLE문을 발행하면 실행 불능 요약 테이블이 대체되고 CREATE TABLE 문은 경로를 리턴합니다(SQLSTATE 01595).

실행 불능 요약 테이블은 SYSCAT.VIEWS 카탈로그 뷰의 VALID 컬럼에 있는 X와 SYSCAT.TABLES 카탈로그 뷰의 STATUS 컬럼에 있는 X로 나타납니다.

- ***특권:*** 테이블이 작성될 때 테이블 정의자에게는 CONTROL 특권이 권한 부여됩니다. 서브테이블이 작성되는 경우, 각 사용자나 그룹이 중간 수퍼테이블에 대해 가지는 SELECT 특권이 권한 준 사용자으로서 테이블 정의자에게 해당 서브테이블에 대한 권한이 자동으로 부여됩니다.
- ***행 크기:*** 테이블의 행에 허용되는 최대 바이트 수는 테이블이 작성된 테이블 공간의 페이지 *크기*에 따라 결정됩니다(*tablspace-name1*). 다음 목록에는 각 테이블 공간 페이지 *크기*에 연관된 행 *크기* 한계 및 컬럼 한계 수가 나와 있습니다.

표 23. 각 테이블 공간 페이지 크기에서의 컬럼 수 및 행 크기의 한계

페이지 *크기*	행 *크기* 한계	컬럼 수 한계
4K	4 005	500
8K	8 101	1 012
16K	16 293	1 012
32K	32 677	1 012

테이블에 대한 실제 컬럼 수는 다음 공식에 의해 더 제한됩니다.

  – 총 컬럼 수 * 8 + LOB 컬럼 수 * 12 + 데이터링크 컬럼 수 * 28 <= 페이지 *크기*에 대한 행 *크기* 한계

- ***바이트 계수:*** 다음 목록에는 널(NULL) 값을 허용하는 않는 컬럼에 대한 데이터 유형별 컬럼 바이트 수가 나와 있습니다. 널(NULL) 값을 허용하는 컬럼의 경우, 바이트 수는 일람표에 표시된 것보다 1이 더 큽니다.

구조화 유형에 기초하여 테이블이 작성되는 경우, 서브테이블이 정의되어 있는지에 관계없이 서브테이블 행을 식별하기 위해 4바이트의 추가 오버헤드가 예약되어 있습니다. 또한, 널(NULL) 입력 불가능으로 정의되어 있는 경우에도, 바이트 계산상 추가 서브테이블 컬럼을 널(NULL) 입력이 가능한 것으로 간주해야 합니다.

데이터 유형	바이트 수
INTEGER	4
SMALLINT	2
BIGINT	8
REAL	4
DOUBLE	8
DECIMAL	(*p*/2)+1의 정수 부분, 여기서, *p*는 정밀도입니다.
CHAR(*n*)	*n*
VARCHAR(*n*)	*n*+4
LONG VARCHAR	24
GRAPHIC(*n*)	*n**2
VARGRAPHIC(*n*)	(*n**2)+4
LONG VARGRAPHIC	24
DATE	4
TIME	3
TIMESTAMP	10
DATALINK(*n*)	*n*+54
LOB types	각 LOB 값은 기본 레코드에 실제 값의 위치를 지시하는 *LOB 설명자*를 갖고 있습니다. 설

명자의 크기는 컬럼에 대해 정의된 최대 길이에 따라 다릅니다. 다음 테이블은 일반적인 크기를 나타냅니다.

최대 **LOB** 길이	**LOB** 설명자 크기
1 024	72
8 192	96
65 536	120
524 000	144
4 190 000	168
134 000 000	200
536 000 000	224
1 070 000 000	256
1 470 000 000	280
2 147 483 647	316

구별 유형 구별 유형의 소스 유형 길이.

참조 유형 참조 유형이 기초한 내장 데이터 유형 길이.

구조화 유형 INLINE LENGTH + 4. INLINE LENGTH는 *column-options* 절의 컬럼에 대해 지정된(또는 명시적으로 계산된) 값입니다.

예

예 1: DEPARTX 테이블 공간에 TDEPT 테이블을 작성합니다. DEPTNO, DEPTNAME, MGRNO 및 ADMRDEPT는 컬럼 이름입니다. CHAR은 컬럼에 문자 데이터가 포함되어 있음을 의미합니다. NOT NULL은 컬럼에 널(NULL) 값을 포함할 수 없음을 의미합니다. VARCHAR은 컬럼에 가변 길이 문자 데이터가 포함됨을 의미합니다. 1차 키는 DEPTNO 컬럼으로 구성됩니다.

```
CREATE TABLE TDEPT
  (DEPTNO   CHAR(3)     NOT NULL,
   DEPTNAME VARCHAR(36) NOT NULL,
   MGRNO    CHAR(6),
   ADMRDEPT CHAR(3)     NOT NULL,
   PRIMARY KEY(DEPTNO))
IN DEPARTX
```

예 2: SHED 테이블 공간에 PROJ 테이블을 작성합니다. PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTAFF, PRSTDATE, PRENDATE 및 MAJPROJ는 컬럼 이름입니다. CHAR은 컬럼에 문자 데이터가 포함되어 있음을 의미합니다.

DECIMAL은 팩된 십진 데이터 컬럼이 포함되어 있음을 의미합니다. 5,2는 다음을 의미합니다. 5는 10진 자리수를 나타내고, 2는 소수점 오른쪽에 있는 자리수를 나타냅니다. NOT NULL은 컬럼에 널(NULL) 값을 포함할 수 없음을 의미합니다. VARCHAR은 컬럼에 가변 길이 문자 데이터가 포함됨을 의미합니다. DATE는 컬럼에 세 부분 형식(년, 월, 일)으로 되어 있는 날짜 정보가 포함됨을 의미합니다.

```
CREATE TABLE PROJ
  (PROJNO   CHAR(6)      NOT NULL,
   PROJNAME VARCHAR(24)  NOT NULL,
   DEPTNO   CHAR(3)      NOT NULL,
   RESPEMP  CHAR(6)      NOT NULL,
   PRSTAFF  DECIMAL(5,2)         ,
   PRSTDATE DATE                 ,
   PRENDATE DATE                 ,
   MAJPROJ  CHAR(6)      NOT NULL)
IN SCHED
```

예 3: 알 수 없는 급여는 0으로 간주하는 EMPLOYEE_SALARY 테이블을 작성하십시오. 테이블 공간이 지정되지 않았으므로, 테이블은 *IN tablespace-name1* 절에 대해 기술된 규칙에 기초하여 시스템에서 선택한 테이블 공간에 작성됩니다.

```
CREATE TABLE EMPLOYEE_SALARY
  (DEPTNO   CHAR(3)      NOT NULL,
   DEPTNAME VARCHAR(36)  NOT NULL,
   EMPNO    CHAR(6)      NOT NULL,
   SALARY   DECIMAL(9,2) NOT NULL WITH DEFAULT)
```

예 4: 총 급여와 마일에 대한 구별 유형을 작성하고 이들을 기본 테이블 공간에 작성되는 테이블의 컬럼에 사용합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 JOHNDOE이고 CURRENT PATH는 기본값(″SYSIBM″, ″SYSFUN″,″JOHNDOE″)이라고 가정합니다.

SALARY값이 지정되어 있지 않으면 이를 0으로 설정해야 하며, LIVING_DIST 값이 지정되어 있지 않으면 이를 1마일로 설정해야 합니다.

```
CREATE DISTINCT TYPE JOHNDOE.T_SALARY AS INTEGER WITH COMPARISONS
CREATE DISTINCT TYPE JOHNDOE.MILES AS FLOAT WITH COMPARISONS
CREATE TABLE EMPLOYEE
   (ID          INTEGER NOT NULL,
    NAME        CHAR (30),
```

```
SALARY      T_SALARY NOT NULL WITH DEFAULT,

LIVING_DIST MILES    DEFAULT MILES(1) )
```

예 5: 이미지와 오디오에 대한 구별 유형을 작성하고 이들을 테이블의 컬럼에 사용합니다. 테이블 공간이 지정되지 않았으므로, *IN 테이블 공간 이름1* 절에 기술된 규칙에 근거하여 시스템이 선택한 테이블 공간에 테이블이 작성됩니다. CURRENT PATH가 기본값이라고 가정합니다.

```
CREATE DISTINCT TYPE IMAGE AS BLOB (10M)
CREATE DISTINCT TYPE AUDIO AS BLOB (1G)
CREATE TABLE PERSON
   (SSN    INTEGER NOT NULL,
    NAME        CHAR (30),
    VOICE  AUDIO,
    PHOTO  IMAGE)
```

예 6: HUMRES 테이블 공간에 EMPLOYEE 테이블을 작성합니다. 테이블에 정의된 제한조건은 다음과 같습니다.

- 부서 번호의 값은 10 - 100 범위 사이여야 합니다.
- 사원의 직위는 'Sales', 'Mgr' 또는 'Clerk' 중 하나가 될 수 있습니다.
- 1986년 이후에 입사한 모든 사원들은 $40,500 이상을 받아야 합니다.

주: 점검 제한조건에 포함된 컬럼이 널(NULL) 입력 가능할 경우, 이 컬럼들은 널(NULL)도 될 수 있습니다.

```
  CREATE TABLE EMPLOYEE
 (ID          SMALLINT NOT NULL,
  NAME        VARCHAR(9),
  DEPT        SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
  JOB         CHAR(5) CHECK (JOB IN ('Sales','Mgr','Clerk')),
  HIREDATE    DATE,
  SALARY      DECIMAL(7,2),
  COMM        DECIMAL(7,2),
  PRIMARY KEY (ID),
  CONSTRAINT  YEARSAL CHECK (YEAR(HIREDATE) > 1986 OR SALARY > 40500)
 )
 IN HUMRES
```

예 7: PAYROLL 테이블 공간에 모두 포함되는 테이블을 작성합니다.

```
CREATE TABLE EMPLOYEE .....
   IN PAYROLL
```

예 8: 데이터 부분은 ACCOUNTING에 포함되고 색인 부분은 ACCOUNT_IDX에 포함되는 테이블을 작성합니다.

```
CREATE TABLE SALARY.....
   IN ACCOUNTING INDEX IN ACCOUNT_IDX
```

예 9: 기본 형식으로 테이블을 작성하고 SQL 변경사항을 기록합니다.

```
CREATE TABLE SALARY1 .....
```

또는

```
CREATE TABLE SALARY1 .....
   DATA CAPTURE NONE
```

예 10: 확장 형식으로 테이블을 작성하고 SQL 변경사항을 기록합니다.

```
CREATE TABLE SALARY2 .....
   DATA CAPTURE CHANGES
```

예 11: SCHED 테이블 공간에 EMP_ACT 테이블을 작성합니다. EMPNO, PROJNO, ACTNO, EMPTIME, EMSTDATE 및 EMENDATE는 컬럼 이름입니다. 테이블에 정의된 제한조건은 다음과 같습니다.

- 모든 행에 있는 컬럼 세트, EMPNO, PROJNO 및 ACTNO에 대한 값은 고유해야 합니다.
- PROJNO의 값은 PROJECT 테이블의 PROJNO 컬럼에 대한 기존 값과 일치해야 하며, 프로젝트가 삭제된 경우, EMP_ACT에서 그 프로젝트를 참조하는 모든 행도 삭제되어야 합니다.

```
CREATE TABLE EMP_ACT
(EMPNO       CHAR(6) NOT NULL,
 PROJNO      CHAR(6) NOT NULL,
 ACTNO       SMALLINT NOT NULL,
 EMPTIME     DECIMAL(5,2),
 EMSTDATE    DATE,
 EMENDATE    DATE,
 CONSTRAINT EMP_ACT_UNIQ UNIQUE (EMPNO,PROJNO,ACTNO),
 CONSTRAINT FK_ACT_PROJ FOREIGN KEY (PROJNO)
                        REFERENCES PROJECT (PROJNO) ON DELETE CASCADE
)
IN SCHED
```

고유 색인 EMP_ACT_UNIQ이 동일한 스키마에 자동으로 작성되어 고유 제한조건을 적용합니다.

예 12: 아이스하키 영예 전당의 유명한 골에 대한 정보를 보유하는 테이블을 작성합니다. 이 테이블에는 골득점한 선수, 득점시 골키퍼, 날짜 및 장소, 설명 등에 대한 정보가 나열될 것입니다. 가능하면, 골 장면의 스틸 사진이나 영상 데이터가 저장되어 있고 경기에 대한 신문 기사가 저장되어 있는 장소를 가리킬 수도 있을 것입니다. 신문 기사는 링크로 연결되어, 삭제하거나 재명명할 수는 없지만 기존의 모든 표시 및 갱신 응용프로그램은 계속 작동되도록 해야 할 것입니다. 스틸 사진과 동영상은 DB2의 완전한 제어하에서 액세스 링크될 것입니다. 링크가 해제될 경우에는 스틸 사진이 복구되어 원래 소유자에게로 돌아갑니다. 동영상 데이터에는 복구 기능이 없으므로, 링크가 해제될 경우에는 삭제됩니다. 설명 컬럼과 세 개의 DATALINK 컬럼이 널(NULL) 입력 가능합니다.

```
CREATE TABLE HOCKEY_GOALS
  ( BY_PLAYER        VARCHAR(30)   NOT NULL,
    BY_TEAM          VARCHAR(30)   NOT NULL,
    AGAINST_PLAYER   VARCHAR(30)   NOT NULL,
    AGAINST_TEAM   VARCHAR(30)   NOT NULL,
    DATE_OF_GOAL     DATE          NOT NULL,
    DESCRIPTION      CLOB(5000),
    ARTICLES         DATALINK  LINKTYPE URL FILE LINK CONTROL MODE DB2OPTIONS,
    SNAPSHOT         DATALINK  LINKTYPE URL FILE LINK CONTROL
                               INTEGRITY ALL
                               READ PERMISSION DB WRITE PERMISSION BLOCKED
                               RECOVERY YES ON UNLINK RESTORE,
    MOVIE            DATALINK  LINKTYPE URL FILE LINK CONTROL
                               INTEGRITY ALL
                               READ PERMISSION DB WRITE PERMISSION BLOCKED
                               RECOVERY NO ON UNLINK DELETE )
```

예 13: EMPLOYEE 테이블에 예외 테이블이 필요하다고 가정하십시오. 다음 명령문을 사용하여 하나 작성할 수 있습니다.

```
CREATE TABLE EXCEPTION_EMPLOYEE AS
  (SELECT EMPLOYEE.*,
     CURRENT TIMESTAMP AS TIMESTAMP,
     CAST ('' AS CLOB(32K)) AS MSG
   FROM EMPLOYEE
   ) DEFINITION ONLY
```

예 14: 표시된 속성을 갖는 다음 테이블 공간이 제공됩니다.

```
TBSPACE            PAGESIZE    USER   USERAUTH
------------------ ----------- ------ --------
DEPT4K                    4096 BOBBY  Y
PUBLIC4K                  4096 PUBLIC Y
DEPT8K                    8192 BOBBY  Y
DEPT8K                    8192 RICK   Y
PUBLIC8K                  8192 PUBLIC Y
```

- RICK이 다음 테이블을 작성하는 경우 이 테이블은 바이트 계수가 4005 미만이므로 테이블 공간 PUBLIC4K에 위치하나, BOBBY가 같은 테이블을 작성하는 경우에는 명시적 권한 부여로 인해 BOBBY가 USE 특권을 가지므로 테이블이 테이블 공간 DEPT4K에 위치합니다.

```
CREATE TABLE DOCUMENTS
   (SUMMARY    VARCHAR(1000),
    REPORT     VARCHAR(2000))
```

- BOBBY가 다음 테이블을 작성하는 경우 이 테이블은 바이트 계수가 4005보다 크고 명시적 권한 부여로 인해 BOBBY가 USE 특권을 가지므로 테이블이 테이블 공간 DEPT8K에 위치합니다. 그러나 DUNCAN이 같은 테이블을 작성하는 경우에는 DUNCAN이 특정 특권을 가지지 않으므로 테이블이 PUBLIC8K에 위치합니다.

```
CREATE TABLE CURRICULUM
   (SUMMARY    VARCHAR(1000),
    REPORT     VARCHAR(2000),
    EXERCISES  VARCHAR(1500))
```

예 15: 구조화 유형 EMP로 정의된 LEAD 컬럼이 있는 테이블을 작성하십시오. LEAD 컬럼에 대해 300 바이트의 INLINE LENGTH를 지정하십시오. 이는 300 바이트내에 들어갈 수 없는 LEAD의 인스턴스는 테이블 외부에 저장됨을 의미합니다(LOB 값의 처리 방법과 유사하게 기본 테이블 행과는 별도로).

```
CREATE TABLE PROJECTS (PID INTEGER,
  LEAD EMP INLINE LENGTH 300,
  STARTDATE DATE,
            ...)
```

예 16: 다섯 개의 컬럼 DEPTNO, DEPTNAME, MGRNO, ADMRDEPT 및 LOCATION을 갖는 테이블 DEPT를 작성하십시오. DEPT 컬럼은 DB2가 그에 대한 값을 항상 생성하도록 식별 컬럼으로 정의됩니다. DEPT 컬럼의 값은 500에서 시작하여 1씩 증가해야 합니다.

```
CREATE TABLE DEPT
   (DEPTNO SMALLINT NOT NULL
      GENERATED ALWAYS AS IDENTITY
      (START WITH 500, INCREMENT BY 1),
    DEPTNAME VARCHAR (36) NOT NULL,
    MGRNO    CHAR(6),
    ADMRDEPT SMALLINT NOT NULL,
    LOCATION CHAR(30))
```

CREATE TABLESPACE

CREATE TABLESPACE문은 데이터베이스 내에 새로운 공간을 작성하여 테이블 공간에 컨테이너를 지정하며, 테이블 공간 정의와 속성을 카탈로그에 기록합니다.

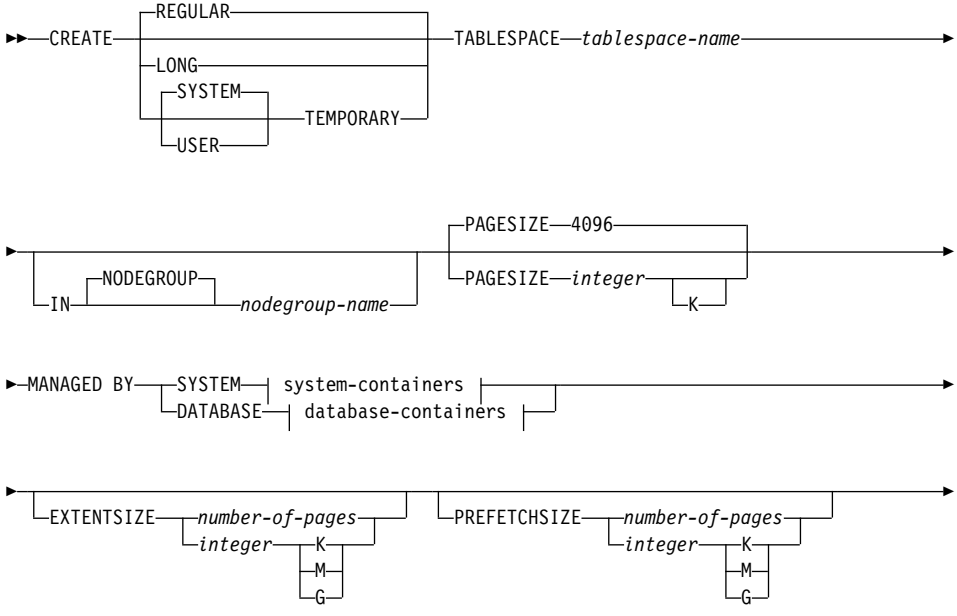
호출

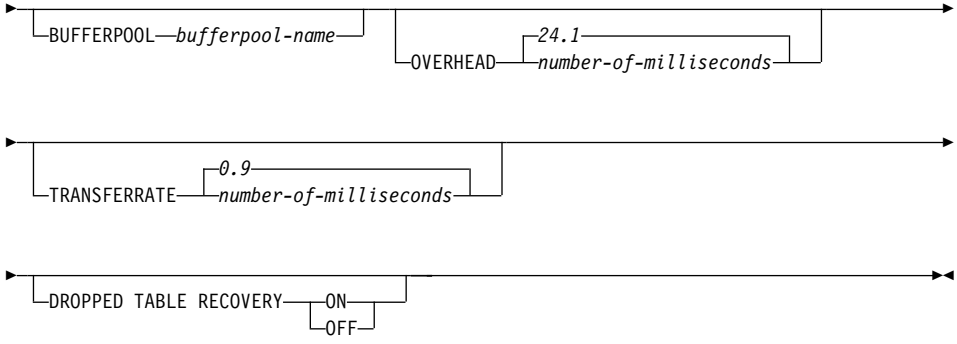
이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

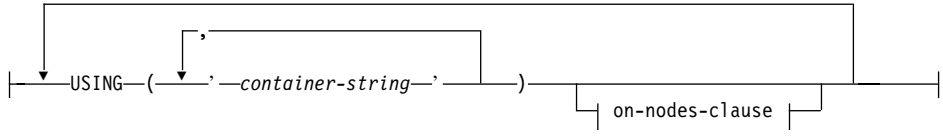
명령문의 권한 부여 ID는 SYSCTRL 또는 SYSADM 권한이 있어야 합니다.

구문

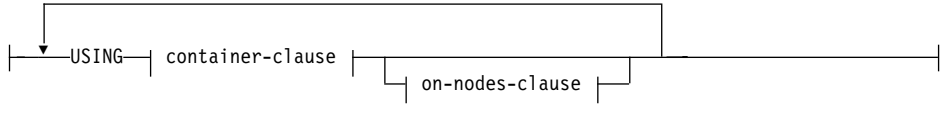




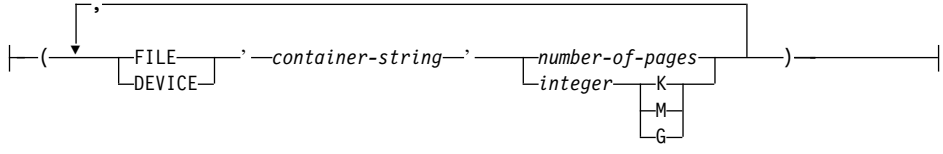
system-containers:



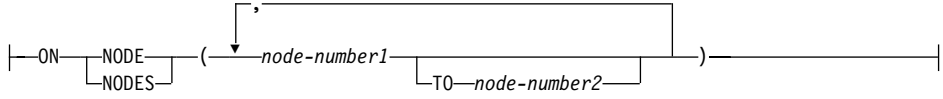
database-containers:



container-clause:



on-nodes-clause:



설명

REGULAR

임시 테이블을 제외한 모든 테이블에 대한 데이터를 저장합니다.

LONG

긴 또는 LOB 테이블 컬럼을 저장합니다. 이것은 또한 저장 구조화 유형 컬럼입니다. 테이블 공간은 DMS 테이블 공간이어야 합니다.

SYSTEM TEMPORARY

임시 테이블을 저장합니다(데이터베이스 관리 프로그램이 정렬 또는 조인과 같은 조작을 수행하는 데 사용하는 작업 영역). 키워드 SYSTEM은 선택적입니다. 하나의 데이터베이스는 임시 테이블만을 그러한 테이블 공간에 저장할 수 있으므로 항상 최소한 하나의 SYSTEM TEMPORARY 테이블 공간을 가져야 합니다. 임시 테이블 공간은 데이터베이스가 작성될 때 자동으로 작성됩니다.

보다 자세한 정보는 *Command Reference*에 있는 CREATE DATABASE를 보십시오.

USER TEMPORARY

선언된 전역 임시 테이블을 저장합니다. 데이터베이스 선언시 사용자 임시 테이블 공간이 없음에 유의하십시오. 선언된 임시 테이블의 정의가 가능하도록 적절한 USE 특권으로 최소한 하나의 사용자 임시 테이블 공간을 작성해야 합니다.

tablespace-name

테이블 공간을 명명합니다. 이 이름은 한 부분의 이름입니다. 이것은 SQL 식별자(일반 또는 분리 식별자)입니다. *tablespace-name*은 카탈로그에 이미 존재하는 테이블 공간은 식별하면 안 됩니다(SQLSTATE 42710). *tablespace-name*은 문자 SYS로 시작하면 안 됩니다(SQLSTATE 42939).

IN NODEGROUP *nodegroup-name*

테이블 공간에 대한 노드 그룹을 지정합니다. 노드 그룹이 존재해야 합니다. SYSTEM TEMPORARY 테이블 공간 작성시 지정할 수 있는 유일한 노드 그룹은 IBMTEMPGROUP뿐입니다. NODEGROUP 키워드는 선택적입니다.

노드 그룹이 지정되지 않은 경우, 기본 노드 그룹(IBMDEFAULTGROUP)이 REGULAR, LONG 및 USER TEMPORARY 테이블 공간에 사용됩니다. SYSTEM TEMPORARY 테이블 공간의 경우, 기본 노드 그룹 IBMTEMPGROUP이 사용됩니다.

PAGESIZE *integer* **[K]**

테이블 공간에 사용되는 페이지 *크기*를 정의합니다. 접미부 K없는 *integer*에 유효한 값은 4 096 or 8 192, 16 384 또는 32 768입니다. 접미부 K가 있는 *integer*에 유효한 값은 4 또는 8, 16 또는 32입니다. 페이지 *크기*가 이러한 값 중의 하나가 아니거나(SQLSTATE 428DE) 테이블 공간이 연관된 버퍼 풀의 페이지 *크기*와 동일하지 않으면 오류가 발생합니다(SQLSTATE 428CB). 기본값은 4 096바이트(4K) 페이지입니다. *정수*와 K 사이에는 공백이 없을 수도 있고 임의의 수의 공백이 들어갈 수도 있습니다.

MANAGED BY SYSTEM

테이블 공간이 시스템 관리 공간(SMS) 테이블 공간임을 지정합니다.

system-containers

SMS 테이블 공간에 대한 컨테이너를 지정합니다.

USING *('container-string',...)*

SMS 테이블 공간에 대해, 테이블 공간이 속하고 그 테이블 공간의 데이터가 저장될 컨테이너를 식별합니다. *container-string*은 240바이트를 초과할 수 없습니다.

각 *container-string*은 절대 또는 상대 디렉토리 이름이 될 수 있습니다. 절대가 아니면, 디렉토리 이름은 데이터베이스 디렉토리에 상대적입니다. 디렉토리 이름의 구성요소가 존재하지 않으면, 데이터베이스 관리 프로그램이 이를 작성합니다. 테이블 공간이 삭제되면, 데이터베이스 관리 프로그램이 작성한 모든 구성요소가 삭제됩니다. 컨테이너 문자열에 의해 식별된 디렉토리가 있는 경우, 그 디렉토리에 파일 또는 서브디렉토리가 들어 있으면 안 됩니다(SQLSTATE 428B2).

*container-string*의 형식은 운영 시스템에 따라 다릅니다. 컨테이너는 운영 시스템에 대해 정상적인 방법으로 지정됩니다. 예를 들어 OS/2 Windows 95 및 Windows NT 디렉토리 경로가 드라이브 이름 및 ":"으로 시작되는 경우, UNIX 시스템에서 경로는 "/"로 시작됩니다.

원격 자원(예: OS/2, Windows 95 및 Windows NT상의 LAN 경로 재지정 드라이브 또는 AIX상의 NFS 마운트 파일 시스템)은 지원되지 않습니다.

on-nodes-clause

파티션된 데이터베이스에서 컨테이너가 작성되는 파티션 또는 파티션들을 지정합니다. 이 절이 지정되지 않으면 컨테이너가 다른 *on-nodes-clause* 절에 명시적으로 지정되지 않은 노드 그룹의 파티션에 작성됩니다. 노드 그룹 IBMTEMPGROUP에 정의된 SYSTEM TEMPORARY 테이블 공간의 경우, *on-nodes-clause* 절이 지정되지 않으면 컨테이너가 데이터베이스에 추가된 모든 새 파티션 또는 노드에도 작성됩니다. 이 절을 지정하는 데 866에서 자세한 내용을 참조하십시오.

MANAGED BY DATABASE

테이블 공간이 데이터베이스 관리 공간(DMS) 테이블 공간임을 지정합니다.

database-containers

DMS 테이블 공간에 대한 컨테이너를 지정합니다.

USING

컨테이너 절을 도입합니다.

container-clause

DMS 테이블 공간에 대한 컨테이너를 지정합니다.

(FILE|DEVICE *'container-string' number-of-pages,...)*

DMS 테이블 공간에 대해 테이블 공간에 속하고 그 테이블 공간의 데이터가 저장될 하나 이상의 컨테이너를 식별합니다. 컨테이너의 유형(FILE 또는 DEVICE) 및 그 *크기*(PAGESIZE 페이지)가 지정됩니다. *크기*는 정수로도 지정될 수 있으며 숫자 뒤에 K(킬로바이트인 경우), M(메가바이트인 경우) 또는 G(기가바이트인 경우)가 올 수 있습니다. 이런 방법으로 지정될 경우, 페이지 *크기*로 나뉘어지는 최소 바이트 수가 컨테이너에 대한 페이지 수를 결정하는 데 사용됩니다. FILE과 DEVICE 컨테이너를 혼합하여 지정할 수 있습니다. *container-string*은 254바이트를 초과할 수 없습니다.

FILE 컨테이너의 경우, *container-string*은 절대 또는 상대 파일 이름이어야 합니다. 파일 이름이 절대가 아니면 데이터베이스 디렉토리에 대한 상대 파일 이름입니다. 디렉토리 이름의 구성요소가 존재하지 않으면, 데이터베이스 관리 프로그램이 이를 작성합니다. 파일이 존재하지 않으면, 데이터베이스 관리 프로그램이 지정하는 *크기*로 작성되어 *초기* 설정됩니다. 테이블 공간이 삭제되면, 데이터베이스 관리 프로그램이 작성한 모든 구성요소가 삭제됩니다.

주: 파일이 존재하면 파일을 대체하고, 지정된 것보다 작으면 확장됩니다. 지정된 것보다 크면, 파일은 잘리지 않습니다.

DEVICE 컨테이너의 경우, *container-string*은 장치 이름이어야 합니다. 장치는 이미 존재하고 있어야 합니다.

모든 컨테이너는 모든 데이터베이스에서 고유해야 합니다. 하나의 컨테이너는 단 하나의 테이블 공간에만 속할 수 있습니다. 컨테이너의 크기는 다를 수 있으나, 컨테이너의 *크기*가 같으면 성능이 최적화됩니다. *container-string*의 정확한 형식은 운영 시스템에 따라 다릅니다. 컨테이너는 운영 시스템에 대해 정상적인 방법으로 지정됩니다. 컨테이너 선언에 대한 관리 안내서에서 좀더 자세한 내용을 참조하십시오.

원격 자원(OS/2, Windows 95 및 Windows NT상의 LAN 경로 재지정 드라이브 또는 AIX상의 NFS 마운트 파일 시스템)은 지원되지 않습니다.

on-nodes-clause

파티션된 데이터베이스에서 컨테이너가 작성되는 파티션 또는 파티션들을 지정합니다. 이 절이 지정되지 않으면, 컨테이너가 다른 *on-nodes-clause* 절에 명시적으로 지정되지 않은 노드 그룹의 파티션에 작성됩니다. 노드 그룹 IBMTEMPGROUP에 정의된 SYSTEM TEMPORARY 테이블 공간의 경우, *on-nodes-clause* 절이 지정되지 않으면 컨테이너가 데이터베이스에 추가된 모든 새 파티션에도 작성됩니다. 이 절을 지정하는 데 866에서 자세한 내용을 참조하십시오.

on-nodes-clause

파티션된 데이터베이스에서 컨테이너가 작성되는 파티션을 지정합니다.

ON NODES

지정된 특정 파티션을 가리키는 키워드. NODE는 NODES와 동의어입니다.

node-number1

특정의 파티션(또는 노드) 번호를 지정합니다.

TO *node-number2*

파티션(또는 노드) 번호의 범위를 지정합니다. *node-number2*의 값은 *node-number1*의 값보다 크거나 같아야 합니다(SQLSTATE 428A9). 지정된 파티션 번호 사이의 모든 파티션은 노드가 테이블 공간의 노드 그룹에 포함되는 경우, 컨테이너가 작성되는 파티션에 포함됩니다.

번호로 지정되는 파티션과 파티션 범위내의 모든 파티션(또는 노드)은 테이블 공간이 정의된 노드 그룹상에 있어야 합니다(SQLSTATE 42729). 파티션 번호는 명령문에 대한 하나의 *on-nodes-clause* 범위 내에서 또는 명시적으로만 나타납니다(SQLSTATE 42613).

EXTENTSIZE *number-of-pages*

다음 컨테이너로 건너뛰기 전에 컨테이너로 쓰여질 PAGESIZE 페이지의 수를 지정합니다. extent 크기 값은 정수로 지정될 수 있으며 숫자 뒤에 K(킬로바이트의 경우), M(메가바이트의 경우) 또는 G(기가바이트의 경우)가 붙습니다. 이런 식으로 지정될 경우, 바이트 수를 페이지 크기로 나눈 최저값은 extent 크기에 대한 값을 결정하는 데 사용됩니다. 데이터베이스 관리 프로그램은 데이터가 저장됨에 따라 컨테이너를 반복적으로 순환합니다.

DFT_EXTENT_SZ 구성 매개변수에 의해 기본값이 제공됩니다.

PREFETCHSIZE *number-of-pages*

데이터 프리페칭이 수행되는 동안 테이블 공간으로부터 읽어 들일 PAGESIZE 페이지의 수를 지정합니다. 프리페치 크기 값은 정수로도 지정할 수 있으며 숫자 뒤에 K(킬로바이트의 경우), M(메가바이트의 경우) 또는 G(기가바이트의 경우)가 붙습니다. 이런 방법으로 지정될 경우, 바이트 수를 페이지 크기로 나눈 최저값은 프리페치 크기에 대한 페이지 수를

결정하는 데 사용됩니다. 프리페칭은 조회에서 참조되기 전에 조회에서 필요한 데이터를 읽어서, I/O 수행시 조회가 지연되지 않도록 합니다.

DFT_PREFETCH_SZ 구성 매개변수에 의해 기본값이 제공됩니다.(다른 모든 구성 매개변수와 마찬가지로 이 구성 매개변수는 *관리 안내서*에서 상세히 설명됩니다.)

BUFFERPOOL *bufferpool-name*

이 테이블 공간의 테이블에 사용된 버퍼 풀의 이름. 버퍼 풀이 존재해야 합니다(SQLSTATE 42704). 지정되지 않으면, 기본 버퍼 풀(IBMDEFAULTBP)이 사용됩니다. 버퍼 풀의 페이지 *크기*는 테이블 공간에 대해 지정된 페이지 *크기*(또는 기본값)와 일치해야 합니다(SQLSTATE 428CB). 테이블 공간의 노드 그룹은 버퍼 풀에 대해 정의되어야 합니다(SQLSTATE 42735).

OVERHEAD *number-of-milliseconds*

I/O 제어기 오버헤드 및 디스크 탐색 및 대기 시간을 밀리초 단위로 지정하는 숫자 문자형 상수(정수, 소수 또는 부동 소수점). 이 숫자는 모든 컨테이너에 대해 같지 않을 경우, 테이블 공간에 속하는 모든 컨테이너에 대한 평균이어야 합니다. 이 값은 조회 최적화시 I/O의 비용을 계산할 때 사용됩니다.

TRANSFERRATE *number-of-milliseconds*

하나의 페이지를 메모리로 읽는 시간을 밀리초 단위로 지정하는 숫자 리터럴 상수(정수, 소수 또는 부동 소수점). 이 숫자는 모든 컨테이너에 대해 같지 않을 경우, 테이블 공간에 속하는 모든 컨테이너에 대한 평균이어야 합니다. 이 값은 조회 최적화시 I/O의 비용을 계산할 때 사용됩니다.

DROPPED TABLE RECOVERY

지정된 테이블 공간의 삭제된 테이블은 ROLLFORWARD 명령의 RECOVER TABLE ON 옵션을 사용하여 복구될 수 있습니다. 이 절은 일반 테이블 공간에 대해서만 지정할 수 있습니다(SQLSTATE 42613). 삭제된 테이블 복구 방법은 *관리 안내서*에 자세히 나와 있습니다.

주

- 정확한 EXTENTSIZE, PREFETCHSIZE, OVERHEAD 및 TRANSFERRATE 값을 결정하는 방법에 대해서는 *관리 안내서*에서 참조하십시오.
- 테이블 공간에 대한 데이터베이스 관리 공간이나 시스템 관리 공간 사이의 선택은 장단점이 있는 근본적인 선택사항입니다. *관리 안내서*에 이러한 장단점에 대한 사항을 참조하십시오.
- 두 개 이상의 TEMPORARY 테이블 공간이 데이터베이스에 존재하면, 사용에서 균형이 이루어지도록 라운드 로빈 형태로 사용됩니다. *관리 안내서*에 둘 이상의 테이블 공간 사용, 재조정 및 EXTENTSIZE, PREFETCHSIZE, OVERHEAD, TRANSFERRATE의 권장 값에 대한 사항을 참조하십시오.
- 파티션된 데이터베이스에서, 두 개 이상의 파티션이 물리적으로 동일한 노드에 상주할 경우, 그러한 파티션에 대해 동일한 장치나 특정 경로를 지정할 수 없습니다(SQLSTATE 42730). 이러한 환경에서는 각 파티션에 대해 고유한 *container-string*을 지정하거나 상대 경로 이름을 사용하십시오.
- SMS 또는 DMS 컨테이너를 작성할 때 컨테이너 문자열 구문에 대한 노드 표현식을 지정할 수 있습니다. 파티션된 데이터베이스 시스템에서 복수의 논리 노드를 사용할 경우, 일반적으로 노드 표현식을 지정합니다. 이는 컨테이너 이름이 노드 전체에서 고유하도록 합니다(데이터베이스 파티션 서버). 표현식을 지정할 경우 노드 번호는 컨테이너 이름의 일부이거나, 추가 인수를 지정할 경우 인수의 결과는 컨테이너 이름의 일부분이어야 합니다.

  인수『 $N』([blank]$N)을 사용하여 노드 표현식을 나타낼 수 있습니다. 인수는 컨테이너 문자열의 맨 끝에 나타나며 다음 양식 중 하나로만 사용될 수 있습니다. 다음 테이블에서 노드 번호는 5로 가정합니다.

표 24. 컨테이너 작성에 필요한 인수는 다음과 같습니다.

구문	예	값
[blank]$N	" $N"	5
[blank]$N+[number]	" $N+1011"	1016
[blank]$N%[number]	" $N%3"	2
[blank]$N+[number]%[number]	" $N+12%13"	4
[blank]$N%[number]+[number]	" $N%3+20"	22

표 24. 컨테이너 작성에 필요한 인수는 다음과 같습니다. (계속)

구문	예	값
주: – %는 계수입니다 – 모든 경우에서, 연산자는 왼쪽에서 오른쪽으로 평가됩니다.		

다음과 같은 예가 있습니다.

예 1:

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
   (device '/dev/rcont $N' 20000)
```

2 노드 시스템에서, 다음과 같은 컨테이너가 사용됩니다.

```
/dev/rcont0  - on NODE 0
/dev/rcont1  - on NODE 1
```

예 2:

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
   (file '/DB2/containers/TS2/container $N+100' 10000)
```

4 노드 시스템에서, 다음과 같은 컨테이너가 사용됩니다.

```
/DB2/containers/TS2/container100  - on NODE 0
/DB2/containers/TS2/container101  - on NODE 1
/DB2/containers/TS2/container102  - on NODE 2
/DB2/containers/TS2/container103  - on NODE 3
```

예 3:

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
('/TS3/cont $N%2','/TS3/cont $N%2+2')
```

2 노드 시스템에서, 다음과 같은 컨테이너가 작성됩니다.

```
/TS3/cont0  - On NODE 0
/TS3/cont2  - On NODE 0
/TS3/cont1  - On NODE 1
/TS3/cont3  - On NODE 1
```

예

예 1: UNIX용 시스템에서 각각 10 000 4K 페이지의 3 장치를 사용하여 정규 DMS 테이블 공간을 작성합니다. I/O 특성을 지정합니다.

```
CREATE TABLESPACE PAYROLL
   MANAGED BY DATABASE
   USING (DEVICE'/dev/rhdisk6' 10000,
      DEVICE '/dev/rhdisk7' 10000,
      DEVICE '/dev/rhdisk8' 10000)
   OVERHEAD 24.1
   TRANSFERRATE 0.9
```

예 2: 64 페이지의 extent *크기*와 32 페이지의 프리페치 *크기*를 가지고 3개의 별도 드라이브상에서 3개의 디렉토리를 사용하여 OS/2 또는 Windows NT상에 정규 SMS 테이블 공간을 작성합니다.

```
CREATE TABLESPACE ACCOUNTING
   MANAGED BY SYSTEM
   USING ('d:\acc_tbsp', 'e:\ecc_tbsp', 'f:\acc_tbsp')
   EXTENTSIZE 64
   PREFETCHSIZE 32
```

예 3: 256 페이지 extent *크기*로 50,000 페이지씩의 2개 파일을 사용하여 Unix 상에 임시 DMS 테이블 공간을 작성합니다.

```
CREATE TEMPORARY TABLESPACE TEMPSPACE2
   MANAGED BY DATABASE
   USING (FILE '/tmp/tempspace2.f1' 50000,
      FILE '/tmp/tempspace2.f2' 50000)
   EXTENTSIZE 256
```

예 4: Unix 파티션된 데이터베이스상의 노드 그룹 ODDNODEGROUP(노드 1,3,5)에 DMS 테이블 공간을 작성합니다. 모든 파티션(또는 노드)에서 10 000 4K 페이지용으로 장치 /dev/rhdisk0를 사용하십시오. 40 000 4K 페이지의 각 파티션용으로 파티션 특정 장치도 지정하십시오.

```
CREATE TABLESPACE PLANS
   MANAGED BY DATABASE
   USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn1hd01' 40000)
   ON NODE (1)
   USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn3hd03' 40000)
   ON NODE (3)
   USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn5hd05' 40000)
   ON NODE (5)
```

CREATE TRANSFORM

CREATE TRANSFORM 문은 구조화 유형 값을 호스트 언어 프로그램 및 외부 함수, 메소드와 교환하는 데 사용되며 그룹 이름으로 식별되는 변환 함수를 정의합니다.

호출

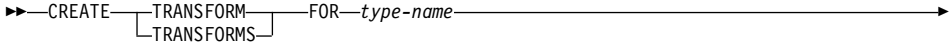
이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

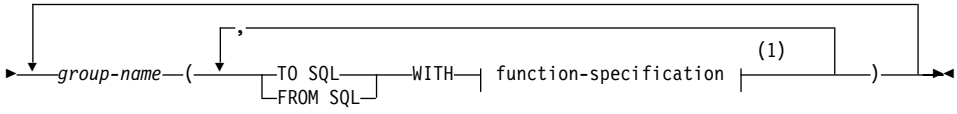
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

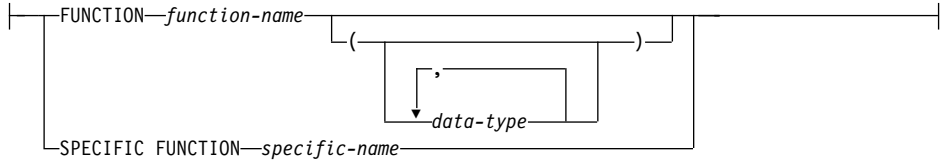
- SYSADM 또는 DBADM 권한
- *type-name*으로 식별되는 유형의 정의자 및 지정된 모든 함수의 정의자.

구문





function-specification:



주:

1 같은 절은 한 번 이상 지정될 수 없습니다.

설명

TRANSFORM 또는 **TRANSFORMS**

하나 이상의 변환 그룹이 정의되고 있음을 나타냅니다. 키워드 버전 중 하나를 지정할 수 있습니다.

FOR *type-name*

변환 그룹이 정의되고 있는 사용자 정의 구조화 유형의 이름을 지정합니다.

동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 *type-name*의 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 사전컴파일/바인드 옵션은 내재적으로 규정화되지 않은 *type-name*의 규정자를 지정합니다. *type-name*은 기존의 사용자 정의 유형의 이름이거나(SQLSTATE 42704), 구조화 유형의 이름이어야 합니다(SQLSTATE 42809). 구조화 유형이나 같은 유형 계층에 있는 다른 구조화 유형의 경우, 제공된 그룹 이름으로 이미 정의되어 있는 변환을 가질 수 없습니다(SQLSTATE 42739).

group-name

TO SQL 및 FROM SQL 함수를 포함하는 변환 그룹의 이름을 지정합니다. 이 이름은 고유하지 않아도 되지만, 이 이름의 변환 그룹(동일한 TO SQL 및 FROM SQL 방향이 정의된)을 지정된 *type-name*에 대해 정의해서는 안됩니다(SQLSTATE 42739). *group-name*은 최대 18자의 SQL 식별자이어야 하며(SQLSTATE 42622) 규정자 접두부를 포함할 수 없습니다(SQLSTATE 42601). *group-name*은 데이터베이스 사용을 위해 예약되어 있으므로 접두부 'sYS'로 시작할 수 없습니다(SQLSTATE 42939).

대부분의 경우 FROM SQL 및 TO SQL 함수 각각을 제공된 그룹에 대해 지정할 수 있습니다(SQLSTATE 42628).

TO SQL

SQL 사용자 정의 구조화 유형 형식으로 값을 변환하는 데 사용되는 세부 함수를 정의합니다. 이 함수는 모든 매개변수를 내장 데이터 유형으로 가져야 하며 리턴된 유형은 *type-name*입니다.

FROM SQL

SQL 사용자 정의 구조화 유형을 나타내는 데이터 유형으로 값을 변환하는 데 사용되는 세부 함수를 정의합니다. 이 함수는 데이터 유형 *type-name*의 매개변수를 하나 가져야 하며 내장 데이터 유형(또는 내장 데이터 유형 집합)을 리턴해야 합니다.

WITH *function-specification*

함수 인스턴스를 지정하는 데 사용할 수 있는 방법은 여러가지가 있습니다.

FROM SQL이 지정된 경우, *function-specification*은 다음 조건에 따른 함수를 식별해야 합니다.

- *type-name* 유형에 대해 하나의 매개변수가 있습니다.
- 리턴 유형이 내장 유형이거나 행이 내장 유형을 모두 가진 컬럼인 경우
- 시그니처는 LANGUAGE SQL 또는 LANGUAGE SQL을 가지는 또 다른 FROM SQL 변환 함수의 사용을 지정합니다.

TO SQL이 지정된 경우, *function-specification*은 다음 조건에 따른 함수를 식별해야 합니다.

- 내장 유형이 있는 모든 매개변수
- 리턴 유형이 *type-name*인지
- 시그니처는 LANGUAGE SQL 또는 LANGUAGE SQL을 가지는 또 다른 TO SQL 변환 함수의 사용을 지정합니다.

메소드(FUNCTION ACCESS로 지정된 경우에도)는 *function-specification*을 통해 변환으로 지정할 수 없습니다. 대신, CREATE FUNCTION 문으로 정의된 함수만이 변환 함수로 작동할 수 있습니다(SQLSTATE 42704 또는 42883).

추가로, 강제 시행되는 경우에도 FROM SQL 함수로부터 리턴된 하나 이상의 내장 유형은 TO SQL 함수의 매개변수인 하나 이상의 내장 유형에 직접 해당되어야 합니다. 이는 이러한 두 함수간의 역관계에 대한 논리적 결과입니다.

FUNCTION *function-name*

특정 함수를 이름으로 식별하고, *function-name*이 있는 함수가 하나인 경우에만 유효합니다. 식별되는 함수는 이에 정의된 매개변수의 수에 관계없이 정의될 수 있습니다.

동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

명명되거나 내재된 스키마에 이 이름의 함수가 없으면, 오류가 발생합니다 (SQLSTATE 42704). 명명된 또는 내재된 스키마에 하나 이상의 특정 함수 인스턴스가 있는 경우, 오류가 발생합니다(SQLSTATE 42725). 표준 함수 선택 알고리즘은 사용하지 않습니다.

FUNCTION *function-name* **(***data-tape,...***)**

사용될 함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 표준 함수 선택 알고리즘은 사용하지 않습니다.

function-name

함수 이름을 지정합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 사전컴파일/바인드 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

(*data-type,...***)**

여기에 지정된 data-type은 해당 위치에 있는 CREATE FUNCTION 문에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형 수와 데이터 유형의 논리적인 병합은 둘다 세부 함수 식별에 사용됩니다.

데이터 유형이 규정되어 있지 않으면, SQL 경로의 스키마를 검색하여 유형 이름을 해석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대한 정밀도 또는 스케일, 길이를 지정하는 것은 필요하지 않습니다. 대신, 데이터 유형 일치를 찾을 때 이러한 속성이 무시됨을 표시하기 위해 빈 괄호 집합을 코드화할 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에, FLOAT()은 사용될 수 없습니다(SQLSTATE 42601). 그러나, 길이, 정밀도 또는 스케일을 적어넣는 경우, 그 값은 CREATE FUNCTION문에 지정된 값과 정확히 일치해야 합니다.

0 <n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치(Matching)는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다. FOR BIT DATA 속성은 같은 목적으로 시그니처의 일부로 고려되지 않음에 유의하십시오. 예를 들어, 시그니처에 지정된 CHAR FOR BIT DATA는 CHAR로만 정의된 함수와 일치합니다.

지정된 이름을 지닌 함수가 명명된 스키마 또는 암시된 스키마에 없는 경우, 오류가 발생합니다(SQLSTATE 42883).

SPECIFIC FUNCTION *specific-name*

함수 작성시 지정되었거나 기본값인 고유 이름을 사용하여 특정 사용자 정의 함수(UDF)를 식별합니다.

동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 사전컴파일/바인드 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. *specific-name*은 명명된 또는 암시된 스키마에서 특정 함수 인스턴스를 식별합니다. 그렇지 않은 경우, 오류가 발생합니다(SQLSTATE42704).

주

- 변환 그룹이 응용프로그램에 지정되지 않은 경우(정적 SQL의 경우 TRANSFORM GROUP 사전 처리 컴파일 또는 바인드 옵션을 사용하거나 동적 SQL의 경우 SET CURRENT DEFAULT TRANSFORM GROUP 문을 사용하여), 응용프로그램이 *type-name*으로 식별된 사용자 정의 구조화 유형을

근거로 호스트 변수를 검색하거나 보내는 중이라면 변환 그룹 'DB2_PROGRAM'의 변환 함수가 (정의된 경우) 사용됩니다. 데이터 유형 *type-name*의 값을 검색하는 경우, FROM SQL 변환을 실행하여 변환 함수가 리턴하는 내장 데이터 유형으로 구조화 유형을 변환합니다. 마찬가지로, 데이터 유형 *type-name*의 값으로 지정될 호스트 변수를 보내는 중이라면 TO SQL 변환을 사용하여 내장 데이터 유형 값을 구조화 유형 값으로 변환합니다. 사용자 정의 변환 그룹이 지정되지 않았거나 'DB2_PROGRAM' 그룹이 정의되지 않은 경우에는(제공된 구조화 유형에 대해) 오류가 발생합니다.

- 구조화 유형 호스트 변수에 대한 내장 데이터 유형 프레젠테이션을 지정할 수 있어야 합니다.
  - 사전 처리 컴파일 명령의 지정된 TRANSFORM GROUP 옵션으로 정의된 대로 (검색 지정 규칙 사용) 구조화 유형에 대한 FROM SQL 변환 함수의 결과로부터
  - 사전 처리 컴파일 명령의 지정된 TRANSFORM GROUP 옵션으로 정의된 대로 (저장영역 지정 규칙 사용) 구조화 유형에 대한 TO SQL 변환 함수의 매개변수로.

  호스트 변수가 적용 가능 변환 함수에 필요한 유형과 호환가능한 지정이 아니라면 오류가 발생합니다(바인드인의 경우 SQLSTATE 42821, 바인드 아웃의 경우 SQLSTATE 42806). 스트링 지정 결과 발생한 오류에 대해서는 111 페이지의 『문자열 지정』에서 참조하십시오.
- 기본 변환 그룹 'DB2_FUNCTION'에서 식별된 변환 함수는 SQL로 작성되지 않은 사용자 정의 함수가 매개변수 또는 리턴 유형으로 데이터 유형 *type-name*을 사용하여 호출될 때마다 사용됩니다. 이는 함수나 메소드가 TRANSFORM GROUP 절을 지정하지 않을 때 적용됩니다. 데이터 유형 *type-name*을 인수로 함수를 호출하는 경우, FROM SQL 변환을 실행하여 구조화 유형을 변환 함수가 리턴하는 내장 데이터 유형으로 변환합니다. 마찬가지로, 함수의 리턴 데이터 유형이 데이터 유형 *type-name*인 경우, TO SQL 변환을 실행하여 외부 함수 프로그램으로부터 리턴된 내장 데이터 유형을 구조화 유형 값으로 변환합니다.
- 구조화 유형에 구조화 유형이기도 한 속성이 포함되는 경우, 연관 변환 함수가 순환적으로 중첩된 모든 구조화 유형을 확장해야 합니다. 이는 변환 함수의 결

과나 매개변수가 주제 구조화 유형의 모든 기본 속성을 나타내는 내장 유형 집합만으로 구성됨을 의미합니다(중첩된 모든 구조화 유형 포함). 중첩된 구조화 유형을 처리하기 위한 변환 함수의 "연쇄" 작용은 없습니다.

- 이 명령문에서 식별된 함수(들)는 이 명령문의 실행시 위에 대략적으로 기술된 규칙에 따라 해석됩니다. 이러한 함수가 후속 SQL문에서 (내재적으로) 사용되면 다른 해석 프로세스를 수행하지 않습니다. 이 명령문에 정의된 변환 함수는 이 명령문에서 해석된 대로 정확하게 기록됩니다.
- 제공된 유형의 속성이나 부속 유형이 작성되거나 삭제될 경우, 사용자 정의 구조화 유형용 변환 함수도 변경해야 합니다.
- 제공된 변환 그룹에 대해 FROM SQL 및 TO SQL 함수를 같은 *group-name* 절, 별개의 *group-name* 절 또는 별개의 CREATE TRANSFORM 문에 지정할 수 있습니다. 먼저 기존의 그룹 정의를 삭제하지 않고는 제공된 FROM SQL 또는 TO SQL 함수 지정을 재정의할 수 없다는 것이 유일한 제한사항입니다. 이를 통해 나중에 먼저 제공된 그룹의 FROM SQL 변환 함수 및 같은 그룹의 해당 TO SQL 변환 함수를 정의할 수 있습니다.

예

예 1: 사용자 정의 구조화 유형 다각형과 C용으로 사용자 정의된 변환 함수 및 Java용으로 특수화된 변환 함수를 연관시키는 두 개의 변환 그룹을 작성하십시오.

```
CREATE TRANSFORM FOR POLYGON
   mystruct1 (FROM SQL WITH FUNCTION myxform_sqlstruct,
              TO SQL WITH FUNCTION myxform_structsql)
   myjava1   (FROM SQL WITH FUNCTION myxform_sqljava,
              TO SQL WITH FUNCTION myxform_javasql )
```

CREATE TRIGGER

CREATE TRIGGER문은 데이터베이스에서 트리거를 정의합니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

트리거 작성시 명령문의 권한 부여 ID가 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

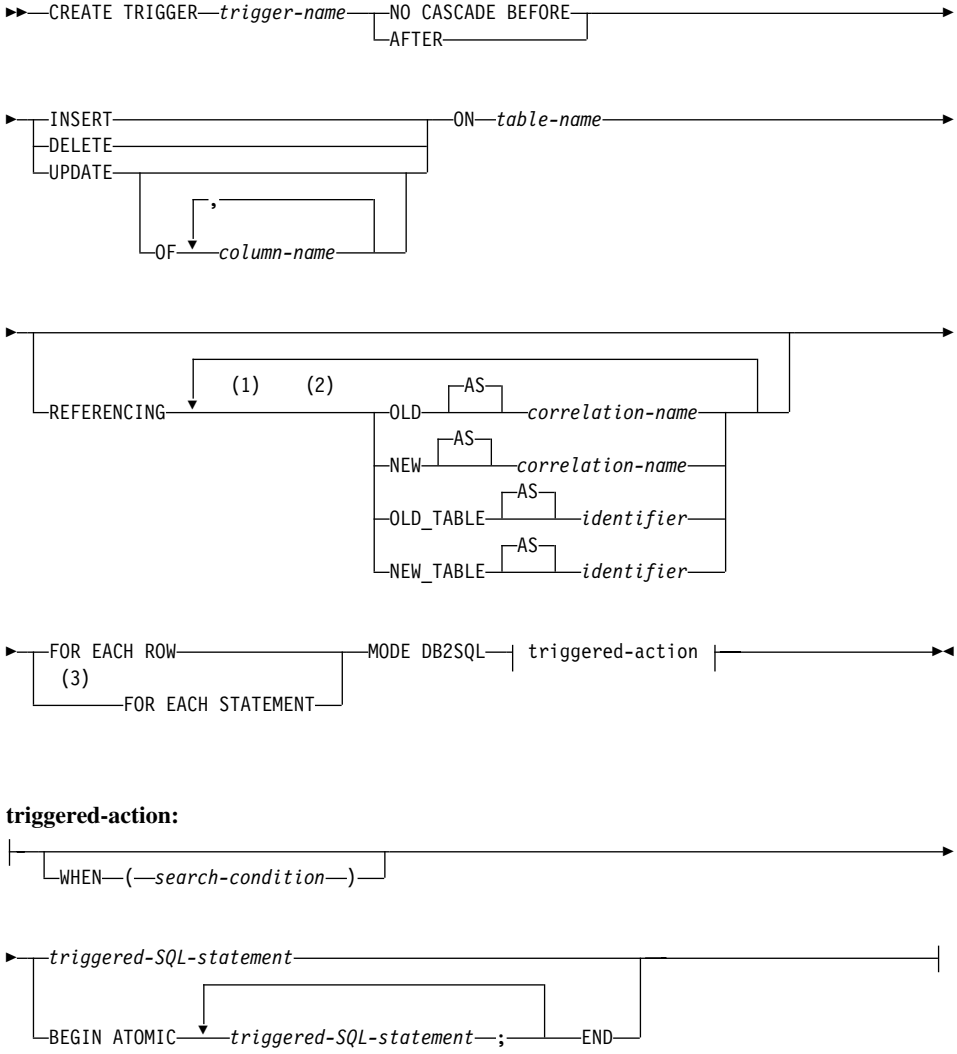
- SYSADM 또는 DBADM 권한
- 트리거가 정의된 테이블에 대한 ALTER 특권 또는 트리거가 정의된 테이블의 스키마와 다음 중 하나에 대한 특권
  - 트리거의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 트리거의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권.

명령문의 권한 부여 ID가 SYSADM 또는 DBADM 권한을 갖고 있지 않은 경우, 트리거가 존재하는 한 (PUBLIC이든 그룹 권한이든) 명령문의 권한 부여 ID가 보유한 권한에는 다음이 모두 포함되어야 합니다.

- 임시 변수나 테이블이 지정된 경우, 트리거가 정의되는 테이블에서의 SELECT 특권
- 트리거되는 조치 상태에서 참조되는 테이블이나 뷰의 SELECT 특권
- 지정된, 트리거된 SQL문을 호출하는 데 필요한 특권

정의자에게 SYSADM 권한이 있으므로 트리거 정의자만이 트리거를 작성할 수 있는 경우, 정의자에게는 트리거 작성용의 명시적인 DBADM 권한이 권한 부여됩니다.

구문



주:

1 OLD와 NEW는 각각 한번만 지정할 수 있습니다.

2 OLD_TABLE와 NEW_TABLE는 AFTER 트리거에 대해서만 각각 한번씩 지정할 수 있습니다.

3 FOR EACH STATEMENT는 BEFORE 트리거에 대해 지정할 수 없습니다.

설명

trigger-name

트리거를 명명합니다. 암시적 또는 명시적 스키마 이름을 포함하는 이름은 카탈로그에 설명된 테이블, 뷰 또는 별명을 식별해서는 안 됩니다. 두 부분의 이름을 지정할 경우, 스키마 이름은 『SYS』로 시작할 수 없습니다(SQLSTATE 42939).

NO CASCADE BEFORE

주제 테이블의 실제 갱신에 의한 변경사항이 데이터베이스에 적용되기 전에 트리거된 연관 조치가 적용됨을 지정합니다. 또한, 트리거의 트리거된 조치로 다른 트리거가 활성화되지 않도록 지정합니다.

AFTER

주제 테이블의 실제 갱신에 의한 변경사항이 데이터베이스에 적용된 후에 트리거된 연관 조치가 적용됨을 지정합니다.

INSERT

트리거와 연관된 트리거 조치가 INSERT 조작이 지정된 기본 테이블에 적용될 때마다 실행되도록 지정합니다.

DELETE

트리거와 연관된 트리거 조치가 DELETE 조작이 지정된 기본 테이블에 적용될 때마다 실행되도록 지정합니다.

UPDATE

트리거와 연관된 트리거 조치가 UPDATE 조작이 지정되거나 암시된 컬럼에 따라 지정된 기본 테이블에 적용될 때마다 실행되도록 지정합니다.

선택적 *column-name* 목록을 지정하지 않으면, 테이블의 모든 컬럼이 지정됨이 암시됩니다. 그러므로, *column-name* 목록을 생략하면, 테이블의 컬럼에 의한 갱신에 의해 트리거가 활성화됩니다.

OF *column-name*,...

지정된 각 *column-name*은 기본 테이블의 컬럼이어야 합니다(SQLSTATE 42703). 트리거가 BEFORE 트리거인 경우, 지정된 *column-name*은 식별 컬럼이 아닌 다른 생성된 컬럼이 아닐 수도 있습니다(SQLSTATE 42989).

*column-name*은 *column-name* 목록에서 두 번 이상 나타나지 않습니다 (SQLSTATE 42711). 트리거는 *column-name* 목록에서 식별되는 컬럼의 갱신에 의해서만 활성화됩니다.

ON *table-name*

트리거 정의의 주제 테이블을 지시합니다. 이름은 기본 테이블을 해석하는 기본 테이블이나 별명을 지정해야 합니다(SQLSTATE 42809). 이름은 카탈로그 테이블 (SQLSTATE 42832), 요약 테이블 (SQLSTATE 42997), 선언된 임시 테이블 (SQLSTATE 42995) 또는 별명 (SQLSTATE 42809)을 지정할 수 없습니다.

REFERENCING

*transition variables*와 *transition tables*에 대한 테이블 이름에 대해 상관 이름을 지정합니다. 상관 이름은 트리거링 SQL 조작에 의해 영향을 받는 행 세트로 특정 행을 식별합니다. 테이블 이름은 영향을 받는 행의 완전한 세트를 식별합니다. 트리거링 SQL 작업에 의해 영향을 받는 각 행은 다음과 같이 지정된 *correlation-names*을 갖는 컬럼을 규정화하여 트리거된 조치에 사용할 수 있습니다.

OLD AS *correlation-name*

트리거링 SQL 조작 이전의 행 상태를 식별하는 상관 이름을 지정합니다.

NEW AS *correlation-name*

트리거링 SQL 조작에 의해, 그리고 이미 실행된 BEFORE 트리거의 SET 문에 의해 수정된 대로 행 상태를 식별하는 상관 이름을 지정합니다.

트리거링 SQL 작업에 의해 영향을 받는 행들의 완전한 집합은 다음과 같이 지정된 임시 테이블 이름을 사용하여 트리거된 조치에 사용할 수 있습니다.

OLD_TABLE AS *identifier*

트리거링 SQL 조작 이전의 영향을 받는 행 세트를 식별하는 임시 테이블 이름입니다.

NEW_TABLE AS *identifier*

트리거링 SQL 조작에 의해, 그리고 이미 실행된 BEFORE 트리거의 SET 문에 의해 수정된 대로 영향을 받는 행을 식별하는 임시 테이블 이름을 지정합니다.

다음 규칙이 REFERENCING절에 적용됩니다.

- OLD 및 NEW 상관 이름과 OLD_TABLE 및 NEW_TABLE 이름 중 어느 것도 동일할 수 없습니다(SQLSTATE 42712).
- 트리거에 대해 단 하나의 OLD 및 NEW *correlation-name*을 지정할 수 있습니다(SQLSTATE 42613).
- 트리거에 대해 단 하나의 OLD_TABLE 및 하나의 NEW_TABLE *identifier*를 지정할 수 있습니다(SQLSTATE 42613).
- OLD *correlation-name*과 OLD_TABLE *identifier*는 트리거 이벤트가 DELETE 조작이거나 UPDATE 조작일 때만 사용할 수 있습니다(SQLSTATE 42898). 조작이 DELETE 조작이면, OLD *correlation-name*은 삭제된 행의 값을 수집저장합니다. 이것이 UPDATE 조작이면, UPDATE 조작 전에 행의 값을 수집저장합니다. OLD_TABLE *identifier*와 영향을 받는 행들의 집합에 동일하게 적용됩니다.
- NEW *correlation-name*과 NEW_TABLE *identifier*는 트리거 이벤트가 INSERT 조작이거나 UPDATE 조작일 경우에만 사용할 수 있습니다(SQLSTATE 42898). 두 조작 모두에서, NEW 값은 원래 조작에서 제공된 대로, 이 시점까지 실행된 BEFORE 트리거에 의해 수정된 대로 행의 새로운 상태를 보관합니다. 이는 NEW_TABLE *identifier* 및 영향받는 행 집합에도 동일하게 적용됩니다.
- OLD_TABLE 및 NEW_TABLE *identifier*는 사전 트리거에 대해 정의할 수 없습니다(SQLSTATE 42898).
- FOR EACH STATEMENT 트리거에 대해 OLD 및 NEW *correlation-name*을 지정할 수 없습니다(SQLSTATE 42899).
- 임시 테이블은 수정할 수 없습니다(SQLSTATE 42807).
- 임시 테이블 컬럼에 대한 총 참조 수와 트리거 조치 내의 임시 변수의 총 수는 테이블 내 컬럼 수에 대한 한계를 초과할 수 없고, 그 길이의 합이 테이블 내 행의 최대 길이를 초과할 수 없습니다(SQLSTATE 54040).
- 각 *correlation-name*과 각 *identifier*의 범위는 전체 트리거 정의입니다.

FOR EACH ROW

트리거된 조치가 트리거링 SQL 조작에 의해 영향을 받는 주제 테이블의 각 행에 대해 한번 적용됩니다.

FOR EACH STATEMENT

트리거된 조치가 전체 명령문에 대해 한번만 적용됨을 지정합니다. 트리거 세분성의 이 유형은 BEFORE 트리거에 대해 지정될 수 없습니다(SQLSTATE 42613). 지정된 경우, UPDATE 또는 DELETE 트리거는 UPDATE 또는 DELETE문을 트리거링하여 영향을 받는 행이 없는 경우에도 활성화됩니다.

MODE DB2SQL

이 절은 트리거의 모드를 지정할 때 사용됩니다. 이것은 현재 지원되는 유일한 유효한 모드입니다.

triggered-action

트리거가 활성화될 때 수행될 조치를 지정합니다. 트리거 조치는 *트리거된 SQL문*의 실행에 대한 선택적 조건에 의해 하나 또는 여러 *트리거된 SQL문*으로 구성됩니다. 주어진 트리거에 대한 트리거 조치에 둘 이상의 *트리거된 SQL문*이 있는 경우, 이는 BEGIN ATOMIC과 END 키워드로 묶이고 세미콜론으로 분리되며 [83] 지정된 순서대로 실행됩니다.

WHEN (*search-condition***)**

참, 거짓 또는 알 수 없음 조건을 지정합니다. *search-condition*은 특정 트리거 조치가 실행되어야 하는지 여부를 판별할 수 있는 능력을 제공합니다.

연관 조치는 지정된 검색 조건이 참으로 평가될 경우에만 수행됩니다. WHEN절이 생략되면, 연관된 *triggered-SQL-statement*이 항상 수행됩니다.

triggered-SQL-statement

트리거가 BEFORE 트리거이면, 트리거된 SQL문은 다음 중 하나여야 합니다(SQLSTATE 42987):

---

83. 이 양식을 명령행 처리기에서 사용하는 경우 명령문 종료 문자는 세미 콜론이 될 수 없습니다. 대체 종료 문자를 지정하는 데 *Command Reference*에서 정보를 참조하십시오.

84. 공통 테이블 표현식은 fullselect 앞에 올 수 있습니다.

- fullselect [84]
- SET transition-variable SQL문
- SIGNAL SQLSTATE문

트리거가 AFTER 트리거이면, 트리거된 SQL문은 다음 중 하나여야 합니다(SQLSTATE 42987):

- INSERT SQL문
- 검색된 UPDATE SQL문
- 검색된 DELETE SQL문
- SIGNAL SQLSTATE문
- fullselect [84]

*트리거된 SQL문*은 정의되지 않은 전이 변수(SQLSTATE 42703) 또는 선언된 임시 테이블을 참조할 수 없습니다(SQLSTATE 42995).

BEFORE 트리거의 *triggered-SQL-statement*은 REFRESH IMMEDIATE로 정의된 요약 테이블을 참조할 수 없습니다(SQLSTATE 42997).

사전 트리거의 *트리거된 SQL문*은 새 전이 변수의 생성된 컬럼(식별 컬럼이 아닌)을 참조할 수 없습니다(SQLSTATE 42989).

주

- 이미 행을 갖고 있는 테이블에 트리거를 추가하면 트리거된 조치가 활성화되지 않습니다. 그러므로, 트리거가 테이블의 데이터에 대해 제한규정을 시행하도록 설계된 경우, 기존 행에서는 이 규정이 충족되지 않을 수도 있습니다.
- 두 트리거에 대한 이벤트가 동시에 발생하는 경우(예를 들어, 이들이 동일한 이벤트, 활동 시간 및 주제 테이블을 갖는 경우), 먼저 작성된 트리거가 먼저 실행됩니다.
- 트리거가 정의된 이후에 주제 테이블에 컬럼이 추가되면, 다음 규칙이 적용됩니다.
  - 트리거가 명시적 컬럼 목록 없이 지정된 UPDATE 트리거인 경우, 새 컬럼에 대한 갱신은 트리거의 활성화를 야기합니다.

  - 컬럼은 이전에 정의된 트리거의 트리거 조치에서 볼 수 없게 됩니다.
  - OLD_TABLE 및 NEW_TABLE 임시 테이블에는 이 컬럼이 포함되지 않습니다. 그러므로, 임시 테이블에서 "SELECT *"의 수행 결과에는 추가된 컬럼이 포함되지 않습니다.
- 컬럼이 트리거 조치에서 참조된 테이블에 추가되면, 새로운 컬럼은 트리거 조치에서 볼 수 없게 됩니다.
- *triggered-SQL-statement*으로 지정된 fullselect의 결과는 트리거 외부나 내부에서 사용할 수 없습니다.
- 연쇄 참조 제한조건의 주기에 포함된 테이블에 정의된 사전 삭제 트리거는 이것이 정의된 테이블이나 참조 무결성 제한조건의 주기 평가시 연쇄적으로 수정된 다른 테이블을 참조하면 안 됩니다. 그러한 트리거의 결과는 데이터에 의존하므로, 일관된 데이터를 작성하지 않을 수도 있습니다.

  가장 간단한 양식에서, 이는 자체 참조 참조 제한사항이 있는 테이블의 사전 삭제 트리거와 CASCADE의 삭제 규칙에서 *triggered-action*의 테이블을 참조하면 안된다는 것을 의미합니다.
- 트리거를 작성하면 특정 패키지가 유효하지 않은 것으로 표시됩니다.
  - 명시적인 컬럼 목록 없이 갱신 트리거를 작성하면, 목표 테이블에서 갱신 사용을 갖는 패키지가 유효하지 않게 됩니다.
  - 컬럼 목록과 함께 갱신 트리거를 작성하면, CREATE TRIGGER문의 *컬럼 이름* 목록에 있는 최소한 하나의 컬럼에서 패키지가 갱신 사용을 갖는 경우 목표 테이블에서 갱신 사용이 없는 패키지만 유효하지 않게 됩니다.
  - 삽입 트리거가 작성되면, 목표 테이블에서 삽입 사용을 갖는 패키지가 유효하지 않게 됩니다.
  - 삭제 트리거가 작성되면. 목표 테이블에서 삭제 사용을 갖는 패키지가 유효하지 않게 됩니다.
- 응용프로그램이 명시적으로 바인딩되거나 리바인딩될 때까지 또는 실행된후 데이터베이스 관리 프로그램에 의해 자동으로 리바인딩될 때까지 패키지는 유효하지 않은 상태로 남습니다.

- *작동 불능 트리거*: 작동 불능 *트리거*는 더 이상 사용할 수 없고 따라서 절대 활성화되지 않는 트리거입니다. 다음의 경우에 트리거는 실행 불능 상태가 됩니다.
  - 트리거 작성자가 트리거를 실행하는 데 필요한 특권이 권한 취소될 경우.
  - 테이블, 뷰 또는 별명과 같이 트리거 조치에 의존하는 오브젝트가 삭제되는 경우.
  - 트리거 조치에 의존하는 뷰가 작동되지 않게 되는 경우.
  - 트리거의 주제 테이블 별명이 삭제된 경우.

  실제 용어에서, 실행 불능 트리거는 DROP 또는 REVOKE문에 대한 연쇄 규칙의 결과로 트리거 정의가 삭제된 트리거입니다. 예를 들어, 뷰가 삭제되면, 그 뷰를 사용하여 정의되고 *triggered-SQL-statement*이 실행 불능 상태가 됩니다.

  트리거가 실행 불능 상태이면, 트리거를 활성화했던 조작을 수행하는 명령문을 갖는 모든 패키지가 유효하지 않은 것으로 표시됩니다. 패키지가 다시 바인딩되면(명시적으로 또는 암시적으로) **실행 불능 트리거는 완전히 무시됩니다**. 마찬가지로, 트리거를 활성화했던 작업을 수행하는 동적 SQL문을 갖는 응용프로그램도 실행 불능 트리거를 완전히 무시합니다.

  트리거 이름은 계속해서 DROP TRIGGER 및 COMMENT ON TRIGGER 문에 지정할 수 있습니다.

  실행 불능 트리거는 실행 불능 트리거의 정의 원문을 사용하는 CREATE TRIGGER문을 발행하여 재작성할 수 있습니다. 이 트리거 정의 원문은 SYSCAT.TRIGGERS의 TEXT 컬럼에 저장됩니다. 실행 불능 트리거를 재작성하기 위해 이를 명시적으로 삭제할 필요가 없음을 주의하십시오. CREATE TRIGGER문을 실행 불능 트리거와 동일한 *trigger-name*과 함께 발행하면, 실행 불능 트리거가 경고와 함께 대체됩니다(SQLSTATE 01595).

  실행 불능 트리거는 SYSCAT.TRIGGERS 카탈로그 뷰의 VALID 컬럼에서 X로 표시됩니다.

- *트리거 실행 중 오류*: 트리거 SQL문 실행 중에 발생하는 오류는 심각한 오류가 아닌 경우 SQLSTATE 09000을 사용하여 리턴됩니다. 오류가 심각하면, 심각한 오류 SQLSTATE가 리턴됩니다. 심각하지 않은 오류에 대한 SQLCA의

SQLERRMC 필드에는 트리거 이름, SQLCODE, SQLSTATE, 그리고 실패한 토큰으로부터 맞출 수 있는 만큼의 토큰이 포함됩니다.

*triggered-SQL-statement*은 SIGNAL SQLSTATE문이 될 수 있거나, RAISE_ERROR 함수를 포함할 수 있습니다. 두 가지 경우 모두, 리턴된 SQLSTATE는 SIGNAL SQLSTATE문 또는 RAISE_ERROR 조건에 지정된 것입니다.

- 아직 존재하지 않는 스키마 이름을 사용하여 트리거를 작성하면, 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- BEFORE 트리거의 실행 전에 데이터베이스 관리 프로그램이 식별 컬럼에 대한 값을 생성합니다. 따라서 생성된 식별 값은 BEFORE 트리거에 가시적입니다.
- 모든 사전 트리거의 실행 후에 데이터베이스 관리 프로그램이 표현식 컬럼에 대한 값을 생성합니다. 따라서 표현식으로 생성된 값은 사전 트리거에 가시적이 아닙니다.
- ***트리거 및 유형화 테이블***: 테이블 계층의 어느 레벨에서도 트리거를 유형화 테이블에 첨부할 수 있습니다. SQL문이 여러 트리거를 활성화하는 경우, 트리거는 유형화 테이블 계층내의 여러 다른 테이블에 첨부되더라도 그 작성 순서 대로 실행됩니다.

  트리거가 활성화되는 경우, 전이 변수(OLD, NEW, OLD_TABLE 및 NEW_TABLE)에 서브테이블 행이 포함될 수도 있습니다. 그러나 첨부된 테이블에 정의된 컬럼만 포함됩니다.

  INSERT, UPDATE 및 DELETE문의 영향:

  – 행 트리거: SQL문을 사용하여 테이블 행을 삽입, 갱신 또는 삭제하는 경우, 이 SQL문은 해당 행을 포함하는 특정 테이블과 이 테이블의 모든 수퍼테이블에 첨부된 행 트리거를 활성화합니다. 이 규칙은 SQL문의 테이블 액세스 방법에 관계없이 항상 적용됩니다. 예를 들어 UPDATE EMP 명령을 발행하는 경우 갱신된 행의 일부가 서브테이블 MGR에 있을 수 있습니다. EMP

행의 경우, EMP 및 그의 수퍼테이블에 첨부된 행 트리거가 활성화됩니다. MGR 행의 경우, MGR 및 그의 수퍼테이블에 첨부된 행 트리거가 활성화됩니다.

– 명령문 트리거: INSERT, UPDATE 또는 DELETE문은 명령문의 영향을 받는 테이블(및 그의 수퍼테이블)에 첨부된 명령문 트리거를 활성화합니다. 이 규칙은 이러한 테이블의 실제 행이 영향을 받는지 여부에 관계없이 항상 적용됩니다. 예를 들어, INSERT INTO EMP 명령에서 EMP 및 그의 수퍼테이블에 대한 명령문 트리거가 활성화됩니다. 다른 예로서, UPDATE EMP 또는 DELETE EMP 명령에서는 서브테이블 행이 갱신되거나 삭제되지 않을지라도 EMP 및 그의 수퍼테이블에 대한 명령문 트리거가 활성화됩니다. 마찬가지로, UPDATE ONLY (EMP) 또는 DELETE ONLY (EMP) 명령은 EMP 및 그의 수퍼테이블에 대한 명령문 트리거는 활성화하나 서브테이블에 대한 명령문 트리거는 활성화하지 않습니다.

DROP TABLE문의 효과: DROP TABLE문은 삭제 중인 테이블에 첨부된 어느 트리거도 활성화하지 않습니다. 그러나 삭제된 테이블이 서브테이블인 경우에는 삭제된 테이블의 모든 행이 수퍼테이블에서 삭제됩니다. 따라서 테이블 T의 경우,

– 행 트리거: DROP TABLE T는 T의 각 행에 대해 T의 모든 수퍼테이블에 첨부된 행 유형 삭제 트리거를 활성화합니다.
– 명령문 트리거: DROP TABLE T는 T에 행이 있는지 여부에 관계없이 T의 모든 수퍼테이블에 첨부된 명령문 유형 삭제 트리거를 활성화합니다.

뷰에 대한 조치: 뷰에 대한 조치로 활성화되는 트리거를 예측하려면, 뷰 정의를 사용하여 해당 조치를 기본 테이블에 대한 조치로 변환하십시오. 예를 들면, 다음과 같습니다.

1. SQL문은 UPDATE V1을 수행합니다. 여기서 V1은 서브뷰 V2를 갖는 유형화 뷰입니다. V1은 기본 테이블 T1을 가지고 V2는 기본 테이블 T2를 가진다고 가정합니다. 명령문은 잠재적으로 T1, T2 및 그의 서브테이블에 있는 행에 영향을 미치므로, T1, T2 및 그의 모든 서브테이블과 수퍼테이블에 대한 명령문 트리거가 활성화됩니다.
2. SQL문은 UPDATE V1을 수행합니다. 여기서 V1은 서브뷰 V2를 갖는 유형화 뷰입니다. V1은 SELECT ... FROM ONLY(T1)으로 정의되고 V2는

SELECT ... FROM ONLY(T2)로 정의된다고 가정합니다. 명령문은 T1과 T2의 서브테이블에 있는 행에 영향을 줄 수 없으므로, T1, T2 및 그의 수퍼테이블에 대한 명령문 트리거는 활성화되나 서브테이블에 대한 명령문 트리거는 활성화되지 않습니다.

3. SQL문은 UPDATE ONLY(V1)을 수행합니다. 여기서 V1은 SELECT ... FROM T1로 정의된 유형화 뷰입니다. 명령문은 잠재적으로 T1과 그의 서브테이블에 영향을 미칠 수 있습니다. 따라서 T1 및 그의 모든 서브테이블과 수퍼테이블에 대한 명령문 트리거가 활성화됩니다.
4. SQL문은 UPDATE ONLY(V1)을 수행합니다. 여기서 V1은 SELECT ... FROM ONLY(T1)로 정의된 유형화 뷰입니다. 이 경우, T1에 서브뷰가 있고 T1에 서브테이블이 있더라도 이 명령문이 영향을 미치는 유일한 테이블은 T1뿐입니다. 따라서 T1 및 그의 수퍼테이블에 대한 명령문 트리거만이 활성화됩니다.

예

예 1: 회사가 관리하는 사원의 수를 자동으로 추적하는 두 개의 트리거를 작성합니다. 트리거는 다음 테이블과 상호 작용합니다.

ID, NAME, ADDRESS, POSITION 컬럼을 갖고 있는 EMPLOYEE 테이블.

NBEMP, NBPRODUCT, REVENUE 컬럼을 갖는 COMPANY_STATS 테이블.

신입 사원이 고용될 때마다, 즉 새로운 행이 EMPLOYEE 테이블에 삽입될 때마다 첫번째 트리거는 사원 수를 증가시킵니다.

```
CREATE TRIGGER NEW_HIRED
AFTER INSERT ON EMPLOYEE
  FOR EACH ROW MODE DB2SQL
UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

두 번째 트리거는 사원이 회사를 그만 둘 때마다, 즉 행이 EMPLOYEE 테이블에서 삭제될 때마다 사원 수를 감소시킵니다.

```
CREATE TRIGGER FORMER_EMP
  AFTER DELETE ON EMPLOYEE
    FOR EACH ROW MODE DB2SQL
  UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1
```

예 2: 부품 레코드가 갱신될 때마다 다음이 점검되고 (필요시) 조치가 취해지도록 하는 트리거를 작성합니다.

남은 양이 최대 재고량의 10% 이내이면, (최대 재고량 - 남은 양)과 같은 영향을 받는 부품에 대해 품목 수만큼 주문하는 선적 요구를 발행합니다.

트리거는 PARTNO, DESCRIPTION, ON_HAND, MAX_STOCKED 및 PRICE 컬럼이 있는 PARTS 테이블과 상호 작용합니다.

ISSUE_SHIP_REQUEST는 해당되는 회사에 추가 부품을 주문하는 양식을 보내는 사용자 정의 함수입니다.

```
   CREATE TRIGGER REORDER
AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
REFERENCING NEW AS N
   FOR EACH ROW MODE DB2SQL
WHEN (N.ON_HAND < 0.10 * N.MAX_STOCKED)
     BEGIN ATOMIC
VALUES(ISSUE_SHIP_REQUEST(N.MAX_STOCKED - N.ON_HAND, N.PARTNO));
     END
```

예 3: 현재 급여의 10 퍼센트 이상 급여 인상을 가져오는 갱신이 발생하는 경우 오류를 발생시키는 트리거를 작성합니다.

```
CREATE TRIGGER RAISE_LIMIT
  AFTER UPDATE OF SALARY ON EMPLOYEE
  REFERENCING NEW AS N OLD AS O
     FOR EACH ROW MODE DB2SQL
  WHEN (N.SALARY > 1.1 * O.SALARY)
        SIGNAL SQLSTATE '75000' ('Salary increase>10%')
```

예 4: 주식 가격에 대한 변경을 기록하고 추적하는 응용프로그램을 고려해봅시다. 데이터베이스에는 CURRENTQUOTE와 QUOTEHISTORY 테이블이 포함됩니다.

```
Tables: CURRENTQUOTE (SYMBOL, QUOTE, STATUS)
        QUOTEHISTORY (SYMBOL, QUOTE, QUOTE_TIMESTAMP)
```

CURRENTQUOTE의 QUOTE 컬럼이 갱신될 때, 새로운 할당량이 시각과 함께 QUOTEHISTORY 테이블에 복사되어야 합니다. 또한, CURRENTQUOTE의 STATUS 컬럼도 다음과 같은 재고 상태를 나타내기 위해 갱신되어야 합니다.

1. 값이 상승할 지의 여부
2. 그 해에 대해 새로운 높은 가격에 있는 지의 여부

3. 값 삭제;
4. 그 해에 대해 새로운 낮은 가격에 있는 지의 여부
5. 값이 안정적인 지의 여부

이를 수행하는 CREATE TRIGGER문은 다음과 같습니다.

- 상태를 설정하기 위한 트리거 정의:

```
CREATE TRIGGER STOCK_STATUS
  NO CASCADE BEFORE UPDATE OF QUOTE ON CURRENTQUOTE
  REFERENCING NEW AS NEWQUOTE OLD AS OLDQUOTE
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
     SET NEWQUOTE.STATUS =
        CASE
           WHEN NEWQUOTE.QUOTE >
                 (SELECT MAX(QUOTE) FROM QUOTEHISTORY
                  WHERE SYMBOL = NEWQUOTE.SYMBOL
                  AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
              THEN 'High'
           WHEN NEWQUOTE.QUOTE <
                 (SELECT MIN(QUOTE) FROM QUOTEHISTORY
                  WHERE SYMBOL = NEWQUOTE.SYMBOL
                  AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
              THEN 'Low'
           WHEN NEWQUOTE.QUOTE > OLDQUOTE.QUOTE
              THEN 'Rising'
           WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE
              THEN 'Dropping'
           WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE
              THEN 'Steady'
        END;
  END
```

- QUOTEHISTORY 테이블에 변경사항을 기록하기 위한 트리거 정의:

```
CREATE TRIGGER RECORD_HISTORY
  AFTER UPDATE OF QUOTE ON CURRENTQUOTE
  REFERENCING NEW AS NEWQUOTE
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
     INSERT INTO QUOTEHISTORY
        VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT TIMESTAMP);
  END
```

CREATE TYPE(구조화)

CREATE TYPE문은 사용자 정의 구조화 유형을 정의합니다. 사용자 정의 구조화 유형에는 제로 또는 그 이상의 속성이 포함될 수 있습니다. 구조화 유형은 속성이 상위 유형으로부터 물려받을 수 있도록 하는 부속 유형이 될 수 있습니다. 명령문의 정상적인 실행으로 속성의 값을 검색하고 갱신하기 위한 메소드가 생성됩니다. 명령문의 정상적인 실행으로 한 컬럼에 사용되는 구조화 유형의 인스턴스를 구성하고 참조 유형 및 표시 유형간의 변환 및 참조 유형의 비교 연산자(=, <>, <, <=, > 및 >=) 지원을 위한 함수도 생성됩니다.

CREATE TYPE문은 사용자 정의 구조화 유형과 함께 사용될 사용자 정의 메소드에 대한 메소드 권장 스펙을 정의하기도 합니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

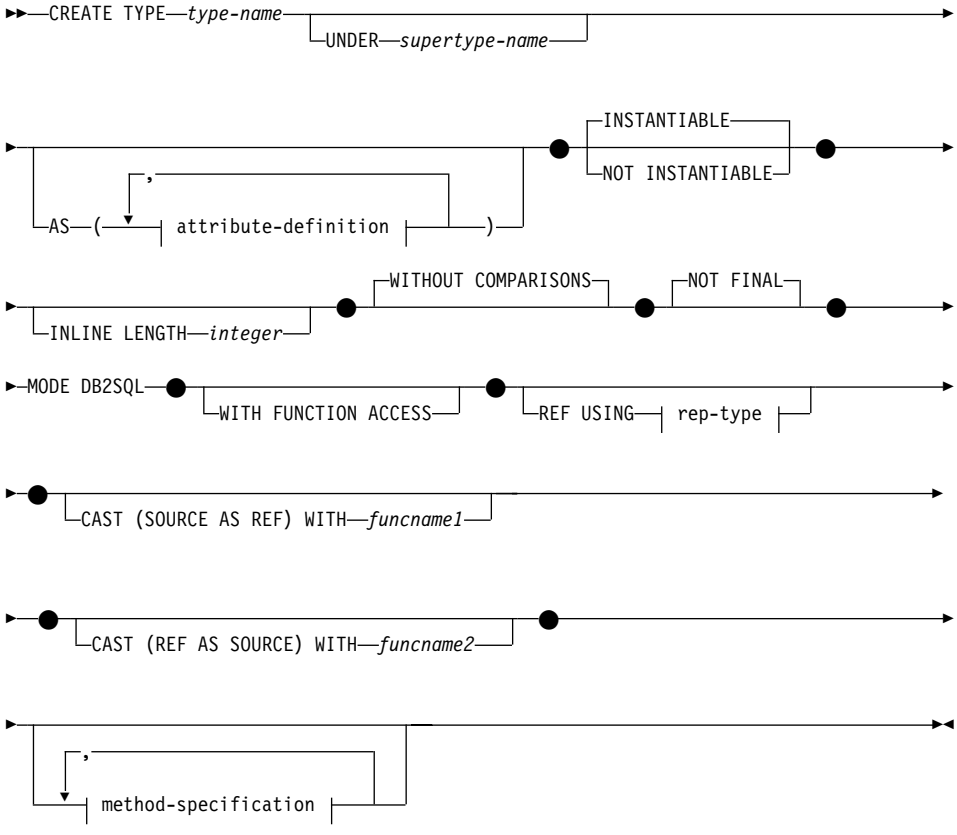
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권에 다음 중 최소한 하나는 포함되어야 합니다.

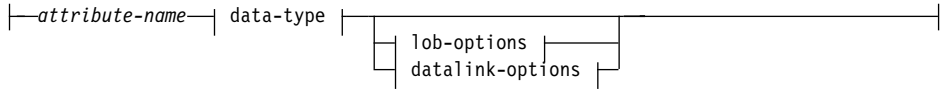
- SYSADM 또는 DBADM 권한
- 유형의 스키마 이름이 기존 스키마를 참조하지 않는 경우 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
- 유형의 스키마 이름이 기존 스키마를 참조하는 경우 스키마에 대한 CREATEIN 특권

UNDER가 지정되고 명령문의 권한 부여 ID가 유형 계층 중 루트 유형 정의자와 동일하지 않을 경우, SYSADM 또는 DBADM 권한이 필요합니다.

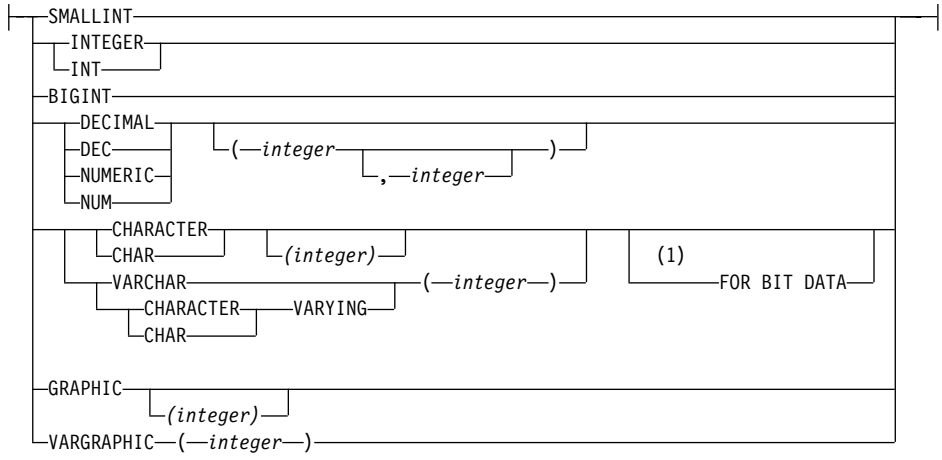
구문



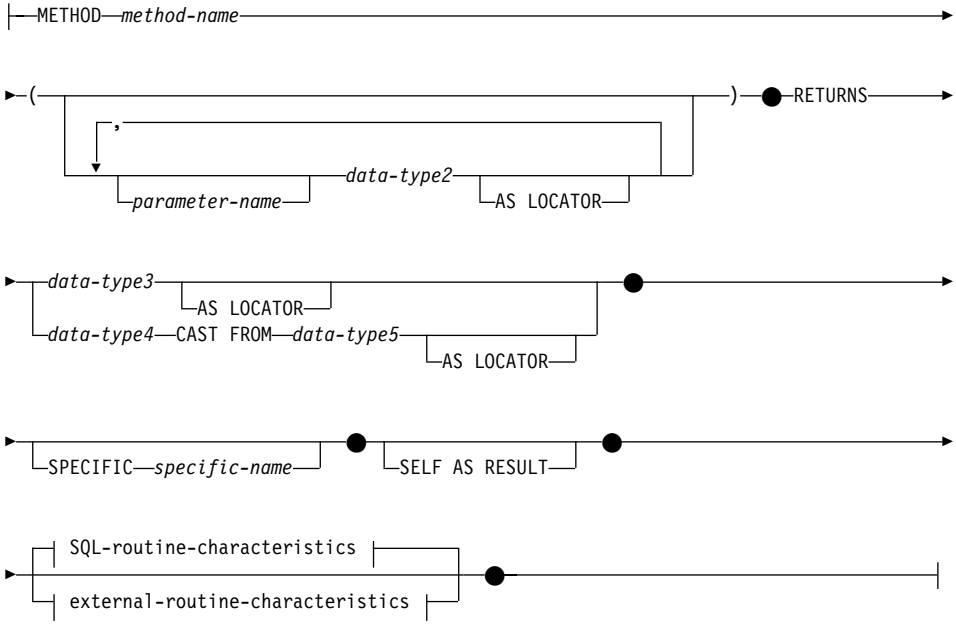
attribute-definition:



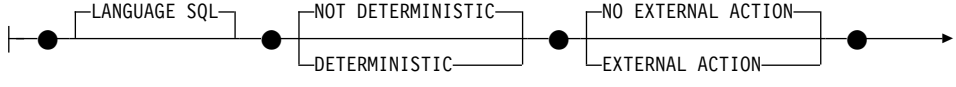
rep-type:

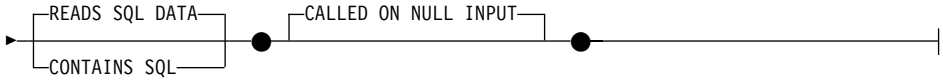


method-specification:

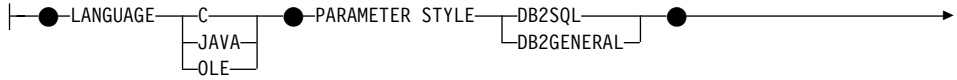


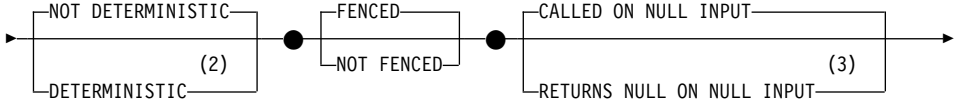
SQL-routine-characteristics:

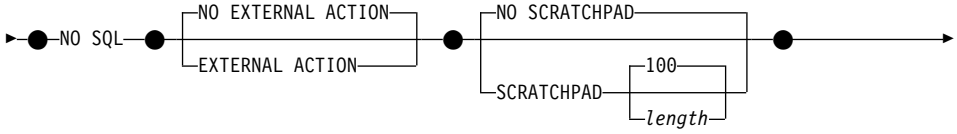


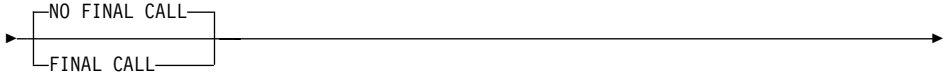


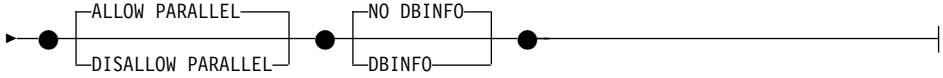
external-routine-characteristics:











주:

1 FOR BIT DATA절은 다른 컬럼 제한조건에 대해 임의의 순서로 지정될 수 있습니다.

2 DETERMINISTIC 대신 NOT VARIANT를 지정할 수 있으며, NOT DETERMINISTIC 대신 VARIANT를 지정할 수 있습니다.

3 CALLED ON NULL INPUT 대신 NULL CALL을 지정할 수 있으며, RETURNS NULL ON NULL INPUT 대신 NOT NULL CALL을 지정할 수 있습니다.

설명

type-name

유형을 명명합니다. 암시적 또는 명시적인 규정자를 포함하여, 이름은 카탈로그에서 이미 설명된 다른 유형(내장 유형, 구조화 유형 또는 구별 유형)을 식별해서는 안 됩니다. 규정되지 않은 이름은 내장 데이터 유형 또는 BOOLEAN 이름과 동일해서는 안 됩니다(SQLSTATE 42918). 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 사전컴파일/바인드 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

스키마 이름(내재된 또는 명시적)은 8바이트를 넘을 수 없습니다(SQLSTATE 42622).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *type-name*으로 사용될 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 등이 이러한 이름이며, 217 페이지의 『기본 술어』에 기술된 비교 연산자도 여기에 포함됩니다.

두 부분의 *type-name*을 지정할 경우, 스키마 이름은 "SYS"로 시작할 수 없습니다. SYS로 시작하면 오류(SQLSTATE 42939)가 발생합니다.

UNDER *supertype-name*

이 구조화 유형이 지정된 *supertype-name* 아래에 있는 부속 유형이 되도록 지정합니다. *supertype-name*은 기존의 구조화 유형을 식별해야 합니다(SQLSTATE 42704). 스키마 이름 없이 *supertype-name*을 지정할 경우, SQL 경로의 스키마를 검색하여 유형이 해석됩니다. 구조화 유형에는 상위 유형의 모든 속성과 그 뒤에 *attribute-definition*에 있는 추가 속성이 포함됩니다.

attribute-definition

구조화 유형의 속성을 정의합니다.

attribute-name

속성의 이름. *attribute-name*은 이 구조화 유형의 다른 속성 또는 상위 유형과 같아서는 안 됩니다(SQLSTATE 42711).

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *attribute-name*으로 사용될 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 등이 이러한 이름이며, 217 페이지의 『기본 술어』에 기술된 비교 연산자도 여기에 포함됩니다.

data-type

속성의 데이터 유형. LONG VARCHAR, LONG VARGRAPHIC이 아닌 800 페이지의 『CREATE TABLE』 아래에 나열된 데이터 유형 중 하나이거나, LONG VARCHAR 또는 LONG VARGRAPHIC에 기초한 구별 유형입니다(SQLSTATE 42601). 데이터 유형은 기존의 데이터 유형을 식별해야 합니다(SQLSTATE 42704). 스키마 이름 없이 *data-type*을 지정할 경우, SQL 경로의 스키마를 검색하여 유형이 해석됩니다. 다양한 데이터 유형에 대한 설명은 800 페이지의 『CREATE TABLE』에 있습니다. 속성 데이터 유형이 참조 유형인 경우, 참조의 목표 유형은 기존의 구조화 유형이어야 하며 그렇지 않으면 이 명령으로 작성됩니다(SQLSTATE 42704).

DATALINK 유형의 속성으로 정의된 구조화 유형은 효과적으로 유형화 테이블 또는 유형화 뷰의 데이터 유형으로서 사용될 수 있습니다(SQLSTATE 01641).

수행시 유형의 인스턴스가 직접 또는 간접적으로 같은 유형 또는 그의 부속 유형 중 하나의 서브유형을 포함하도록 허용하는 유형 정의를 금하기 위해 그의 속성 유형 중 하나가 직접 또는 간접적으로 스스로를 사용하도록 유형을 정의할 수 없습니다(SQLSTATE 428EP). 102 페이지의 『구조화 유형』에서 자세한 설명을 참조하십시오.

lob-options

LOB 유형과 연관된 옵션(또는 LOB 유형에 기초한 구별 유형)을 지정합니다. *lob-options*에 대한 800 페이지의 『CREATE TABLE』에서 자세한 설명을 참조하십시오.

datalink-options

DATALINK 유형과 연관된 옵션(또는 DATALINK 유형에 기초한 구별

유형)을 지정합니다. *datalink-options*에 대한 800 페이지의 『CREATE TABLE』에서 자세한 설명을 참조하십시오.

DATALINK 유형에 대해 옵션이 지정되어 있지 않거나 구별 유형 소스가 DATALINK인 경우, LINKTYPE URL 옵션과 NO LINK CONTROL 옵션이 기본값입니다.

INSTANTIABLE 또는 NOT INSTANTIABLE

구조화 유형의 인스턴스를 작성할 수 있는지 여부를 판별합니다. 다음은 인스턴스 작성 가능한 구조화 유형이 내포하는 사항입니다.

- 인스턴스 작성 가능하지 않은 유형에 대한 구성자 함수가 생성되지 않습니다.
- 인스턴스 작성 가능하지 않은 유형은 테이블이나 뷰의 유형으로 사용할 수 없습니다(SQLSTATE 428DP).
- 인스턴스 작성 가능하지 않은 유형은 컬럼의 유형으로 사용할 수 있습니다(인스턴스 작성 가능한 부속 유형의 널(NULL) 값 또는 인스턴스만을 컬럼에 삽입할 수 있습니다).

인스턴스 작성 가능하지 않은 유형의 인스턴스를 작성하려면 인스턴스 작성 가능 부속 유형을 작성해야 합니다. NOT INSTANTIABLE이 지정되지 않으면 새 유형의 인스턴스를 작성할 수 없습니다.

INLINE LENGTH *integer*

이 옵션은 테이블 행에 값의 나머지 부분을 인라인 저장하기 위한 구조화 유형 컬럼 인스턴스의 최대 *크기*(바이트 단위)를 나타냅니다. 지정된 인라인 길이보다 긴 구조화 유형이나 그이 부속 유형의 인스턴스는 LOB 값의 처리 방법과 유사하게 기본 테이블 행과는 별도로 저장됩니다.

지정된 INLINE LENGTH가 새로 작성된 유형(32바이트+속성당 10바이트)에 대한 구성자 함수의 결과 *크기*보다 작고 292바이트보다 작으면 오류가 발생합니다(SQLSTATE 429B2). 속성 수에는 해당 유형의 상위 유형에서 계승된 모든 속성이 포함됨에 유의하십시오.

유형에 대한 INLINE LENGTH는 지정되었거나 기본값이거나 관계없이 구조화 유형을 사용하는 컬럼의 기본 인라인 길이입니다. CREATE TABLE 수행 시 이 기본값에 겹쳐쓸 수 있습니다.

INLINE LENGTH는 구조화 유형이 유형화 테이블의 유형으로 사용되는 경우에는 아무 의미도 없습니다.

구조화 유형의 기본 INLINE LENGTH는 시스템에 의해 계산됩니다. 아래 제공된 공식에서는 다음 항목이 사용됩니다.

short attribute
: 다음 데이터 유형 중 하나를 갖는 속성을 참조합니다. SMALLINT, INTEGER, BIGINT, REAL, DOUBLE, FLOAT, DATE 또는 TIME. 이러한 유형을 기초로 구별 유형이나 참조 유형도 포함됩니다.

non-short attribute
: 이러한 데이터 유형을 기초로 나머지 데이터 유형이나 구별 유형의 속성을 참조합니다.

시스템은 다음과 같이 기본 인라인 길이를 계산합니다.

1. 다음 공식을 사용하여 short가 아닌 속성에 대한 추가 공간 요구사항을 결정하십시오.

   *space_for_non_short_attribute*s = SUM(*attributelength* + n)

   n은 다음과 같이 정의됩니다.

   - 중첩된 구조화 유형 속성의 경우 0 바이트
   - 비 LOB 속성의 경우 2 바이트
   - LOB 속성의 경우 9 바이트

   *attributelength*는 901 페이지의 표25에 나와 있는 대로 속성에 대해 지정된 데이터 유형을 기초로 합니다.

2. 다음 공식을 사용하여 총 기본 인라인 길이를 계산하십시오.

   default_length(*structured_type*) = (*number_of_attributes* * 10) + 32 + *space_for_non-short_attributes*

   *number_of_attributes*는 그의 상위 유형에서 계승된 속성을 포함하여 구조화 유형에 대한 총 속성 수입니다. 그러나 *number_of_attributes*에는 *structured_type*의 부속 유형에 대해 정의된 속성이 포함되지 않습니다.

표 25. 속성 데이터 유형에 대한 바이트 수

<table>
<tr><td>데이터 유형 속성</td><td colspan="2">바이트 수</td></tr>
<tr><td>DECIMAL</td><td colspan="2">(p/2)+1의 정수 부분. 여기서 p는 정밀도입니다.</td></tr>
<tr><td>CHAR (n)</td><td colspan="2">n</td></tr>
<tr><td>VARCHAR (n)</td><td colspan="2">n</td></tr>
<tr><td>GRAPHIC (n)</td><td colspan="2">n * 2</td></tr>
<tr><td>VARGRAPHIC (n)</td><td colspan="2">n * 2</td></tr>
<tr><td>TIMESTAMP</td><td colspan="2">10</td></tr>
<tr><td>DATALINK(n)</td><td colspan="2">n + 54</td></tr>
<tr><td rowspan="12">LOB 유형</td><td colspan="2">각 LOB 속성은 실제 값의 위치를 가리키는 구조화 유형 인스턴스에 LOB 설명자를 가집니다. 설명자 크기는 LOB 속성에 정의된 최대 길이에 따라 변합니다.</td></tr>
<tr><td>최대 LOB 길이</td><td>LOB 설명자 크기</td></tr>
<tr><td>1 024</td><td>72</td></tr>
<tr><td>8 192</td><td>96</td></tr>
<tr><td>65 536</td><td>120</td></tr>
<tr><td>524 000</td><td>144</td></tr>
<tr><td>4 190 000</td><td>168</td></tr>
<tr><td>134 000 000</td><td>200</td></tr>
<tr><td>536 000 000</td><td>224</td></tr>
<tr><td>1 070 000 000</td><td>256</td></tr>
<tr><td>1 470 000 000</td><td>280</td></tr>
<tr><td>2 147 483 647</td><td>316</td></tr>
<tr><td>구별 유형</td><td colspan="2">구별 유형의 소스 유형 길이</td></tr>
<tr><td>참조 유형</td><td colspan="2">참조 유형을 기초로 한 내장 데이터 유형 길이.</td></tr>
<tr><td>구조화 유형</td><td colspan="2">inline_length(attribute_type)</td></tr>
</table>

WITHOUT COMPARISONS

구조화 유형 인스턴스에 대해 지원되는 비교 함수가 없음을 나타냅니다.

NOT FINAL

구조화 유형이 상위 유형으로서 사용될 수 있음을 나타냅니다.

MODE DB2SQL

이 절은 필수이며 이 유형에 대한 구성자 함수의 직접 호출을 허용합니다.

WITH FUNCTION ACCESS

이 유형 및 부속 유형의 모든 메소드(차후 작성된 메소드 포함)는 함수 표시법을 사용하여 액세스할 수 있음을 나타냅니다. 이 절은 구조화 유형 계층의 루트 유형에 대해서만 지정할 수 있습니다(UNDER절은 지정되지 않습니다)(SQLSTATE 42613). 메소드 호출 표기법보다는 이 표기법 양식을 선호하는 응용프로그램의 함수 표기법 사용이 가능하도록 이 절이 제공됩니다.

REF USING *rep-type*

이 구조 유형과 이것의 모든 서브유형에 대한 표현(기초가 되는 데이터 유형)으로서 사용된 내장 데이터 유형을 정의합니다. 이 절은 구조 유형 계층의 루트 유형에 대해서만 지정될 수 있습니다(UNDER 절이 지정되지 않음)(SQLSTATE 42613). *rep-type*은 LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK 또는 구조화 유형일 수 없으며 255바이트 이하여야 합니다(SQLSTATE 42613).

구조 유형 계층의 루트 유형에 대해 이 절이 지정되지 않으면, REF USING VARCHAR(16) FOR BIT DATA를 가정합니다.

CAST (SOURCE AS REF) WITH *funcname1*

*rep-type*의 데이터 유형을 가진 값을 이 구조화 유형의 참조 유형으로 변환하는 시스템 생성 함수의 이름을 정의합니다. 스키마 이름은 *funcname1*의 일부로서 지정할 수 없습니다(SQLSTATE 42601). 유형변환 함수는 구조화 유형과 같은 스키마에서 생성됩니다. 절이 지정되지 않는 경우, *funcname1*에 대한 기본값은 *type-name*(구조화 유형의 이름)입니다. *funcname1(rep-type)*과 일치하는 함수 시그니처가 같은 스키마에 있을 수 없습니다(SQLSTATE 42710).

CAST (REF AS SOURCE) WITH *funcname2*

이 구조화 유형에 대한 참조 유형을 *rep-type*의 데이터 유형으로 변환하는 시스템 생성 함수의 이름을 정의합니다. 스키마 이름은 *funcname2*의 일부분으로 지정할 수 없습니다(SQLSTATE 42601). 유형변환 함수는 구조화 유형과 같은 스키마에서 생성됩니다. 절이 지정되지 않는 경우, *funcname2*에 대한 기본값은 *rep-type*(표현 유형의 이름)입니다.

method-specification

이 유형에 대한 메소드를 정의합니다. 메소드는 실제로 CREATE METHOD 문이 있는 내용이 제공될 때까지 사용할 수 없습니다(SQLSTATE 42884).

method-name

정의되고 있는 메소드의 이름입니다. 규정화되지 않은 SQL 식별자여야 합니다(SQLSTATE 42601). 메소드 이름은 CREATE TYPE에 사용된 스키마와 함께 내재적으로 규정화됩니다.

술어에서 키워드로 사용되는 많은 이름은 시스템용으로 예약되어 있으므로 *method-name*으로 사용될 수 없습니다(SQLSTATE 42939). SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH 등이 이러한 이름이며 217 페이지의 『기본 술어』에 기술된 비교 연산자도 여기에 포함됩니다.

일반적으로 시그니처에 차이가 있다면 둘 이상의 메소드에 대해 같은 이름을 사용할 수 있습니다.

parameter-name

매개변수 이름을 식별합니다. SELF(메소드의 내재적 주제 매개변수 이름)일 수 없습니다(SQLSTATE 42734). 메소드가 SQL 메소드인 경우, 모든 매개변수가 이름을 가져야 합니다(SQLSTATE 42629).

data-type2

각 매개변수의 데이터 유형을 지정합니다. 메소드가 수신할 각 매개변수에 대해 목록의 한 항목을 지정해야 합니다. 내재된 SELF 매개변수를 포함하여 매개변수는 90개를 초과할 수 없습니다. 이 한계를 초과하면 오류가 발생합니다(SQLSTATE 54023).

CREATE TABLE문의 컬럼 유형으로 지정할 수 있고 메소드를 작성하는 데 사용 중인 언어와 관련 있는 SQL 데이터 유형 권장 스펙 및 약어를 지정할 수 있습니다. 사용자 정의 함수 및 메소드에 대해 SQL 데이터 유형 및 호스트 언어 데이터 유형간의 맵핑에 대해 자세히 알려면 응용프로그램 개발 안내서의 언어 관련 절을 참조하십시오.

주: 질문의 SQL 데이터 유형이 구조화 유형이라면 호스트 언어 데이터 유형에 맵핑되는 기본값이 없습니다. 사용자 정의 변환 함수를 사용하여 구조화 유형과 호스트 언어 데이터 유형간의 맵핑을 작성해야 합니다.

DECIMAL (및 NUMERIC)은 LANGUAGE C와 OLE에 대해 유효하지 않습니다(SQLSTATE 42815). DECIMAL 사용 대안에 대해서는 *응용프로그램 개발 안내서*에 나와 있습니다.

REF를 지정할 수 있으나 정의된 범위는 가지지 않습니다. 메소드 내용에서 먼저 범위를 가지도록 변환해야만 참조 유형을 경로 표현식에 사용할 수 있습니다. 마찬가지로, 먼저 범위를 가지도록 변환해야만 메소드가 리턴하는 참조를 경로 표현식에 사용할 수 있습니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 기초한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 실제 값이 아닌 LOB 위치 지정자가 메소드로 전달됨을 나타냅니다. 그러면 메소드에 전달된 바이트 수가 크게 절약되고 특히 값의 몇몇 바이트만이 실제 메소드와 관련되는 경우 성능도 향상될 수 있습니다. LOB 위치 지정자의 사용에 대해서는 *응용프로그램 개발 안내서*에 설명되어 있습니다.

LOB 또는 LOB에 기초한 구별 유형이 아닌 다른 유형에 대해 AS LOCATOR가 지정된 경우 오류가 발생합니다(SQLSTATE 42601).

메소드가 분리되거나 LANGUAGE가 SQL인 경우에는 AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

RETURNS

이 필수 절은 메소드의 결과를 식별합니다.

data-type3

메소드 결과의 데이터 유형을 지정합니다. 이 경우 정확하게 같은 고려사항이 data-type2 하에서 위에 기술된 메소드의 매개변수에 적용됩니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 실제 값이 아닌 LOB 위치 지정자가 메소드로부터 전달됨을 나타냅니다.

LOB 또는 LOB에 기초한 구별 유형이 아닌 다른 유형에 대해 AS LOCATOR가 지정된 경우 오류가 발생합니다(SQLSTATE 42601).

메소드가 분리되거나 LANGUAGE가 SQL인 경우에는 AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

data-type4 **CAST FROM** *data-type5*

메소드 결과의 데이터 유형을 지정합니다.

이 절은 메소드 코드로 리턴된 데이터 유형에서 호출 명령문으로 다른 데이터 유형을 리턴하는 데 사용됩니다. *data-type5*는 *data-type4* 매개변수에 유형변환할 수 있어야 합니다. 유형변환할 수 없을 경우, 오류가 발생합니다(SQLSTATE 42880).

*data-type4*에 대한 길이, 정밀도 또는 스케일은 *data-type5*로부터 추론될 수 있으므로, *data-type4*에 대해 지정된 매개변수 작성 유형의 길이, 정밀도 또는 스케일을 지정할 필요가 없습니다(아직은 허용됨). 대신 빈 괄호를 사용할 수 있습니다(예: VARCHAR()). 매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에, FLOAT()은 사용될 수 없습니다(SQLSTATE 42601).

구별 유형은 data-type5에 지정된 유형으로 유효하지 않습니다(SQLSTATE 42815).

변환 조작은 변환 오류를 발생시킬 수 있는 런타임 검사에도 종속됩니다.

AS LOCATOR

LOB 유형 또는 LOB 유형에 근거한 구별 유형의 경우, AS LOCATOR절을 추가할 수 있습니다. 이는 실제 값이 아닌 LOB 위치 지정자가 메소드로부터 전달됨을 나타냅니다.

LOB 또는 LOB에 기초한 구별 유형이 아닌 다른 유형에 대해 AS LOCATOR가 지정된 경우 오류가 발생합니다(SQLSTATE 42601).

메소드가 분리되거나 LANGUAGE가 SQL인 경우에는 AS LOCATOR절을 지정할 수 없습니다(SQLSTATE 42613).

SPECIFIC *specific-name*

정의 중인 메소드의 인스턴스에 대한 고유한 이름을 제공합니다. 이 고유 이름은 메소드 내용을 작성하거나 메소드를 삭제하는 경우 사용할 수 있습니다. 메소드를 호출하는 데는 사용할 수 없습니다. *specific-name*의 규정화되지 않은 양식은 SQL 식별자(최대 길이가 18인)입니다. 규정화 양식은 다음에 마침

표와 SQL 식별자가 오는 스키마 이름입니다. 내재된 또는 명시적 규정자를 포함하는 이름은 응용프로그램 서버(AS)에 있는 다른 고유 메소드 이름을 식별할 수 없습니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42710).

*specific-name*은 기존의 *method-name*과 같을 수 있습니다.

규정자가 지정되지 않으면 *type-name*에 사용된 규정자가 사용됩니다. 규정자가 지정되면, *type-name*의 명시적 또는 내재된 규정자와 같아야 하며 그렇지 않으면 오류가 발생합니다(SQLSTATE 42882).

*specific-name*을 지정하지 않으면, 데이터베이스 관리 프로그램에 의해 고유한 이름이 생성됩니다. 고유한 이름은 문자 시간소인이 뒤에 오는 SQL입니다(SQLyymmddhhmmssxxx).

SELF AS RESULT

이 메소드를 유형 보존 메소드로 식별합니다. 이는 다음을 의미합니다.

- 선언된 리턴 유형은 선언된 주제 유형과 같아야 합니다(SQLSTATE 428EQ).
- SQL문이 컴파일되고 유형 유지 메소드로 해석되는 경우, 메소드 결과의 정적 유형은 주제 인수의 정적 유형과 같습니다.
- 메소드는 결과의 동적 유형이 주제 인수의 동적 유형과 같은 방법으로 구현해야 하며 (SQLSTATE 2200G) 그 결과는 널(NULL)이 아닐 수도 있습니다(SQLSTATE 22004).

SQL-routine-characteristics

CREATE METHOD를 사용하여 이 유형에 대해 정의될 메소드 내용의 특성을 지정합니다.

LANGUAGE SQL

이 절은 단일 RETURN문을 통해 메소드가 SQL로 작성됨을 나타내는 데 사용됩니다. 메소드 내용은 CREATE METHOD문을 사용하여 지정됩니다.

NOT DETERMINISTIC 또는 DETERMINISTIC

이 선택적 절은 메소드가 항상 주어진 인수 값에 대해 같은 결과를 리턴하는지 여부(DETERMINISTIC) 또는 메소드가 결과에 영향을 주는 몇몇 상태 값에 따라 달라지는지 여부(NOT DETERMINISTIC)를 지정합니다. 즉, DETERMINISTIC 메소드는 항상 동일한 입력으로 계속되는 호출로

부터 같은 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC을 지정하여 방지할 수 있습니다. 메소드 내용이 특수 레지스터를 액세스하거나 다른 비결정 루틴을 호출하는 경우 NOT DETERMINISTIC을 명시적으로 또는 내재적으로 지정해야 합니다(SQLSTATE 428C2).

NO EXTERNAL ACTION 또는 EXTERNAL ACTION

이 선택적 절은 메소드가 데이터베이스 관리 프로그램에서 관리하지 않는 오브젝트의 상태를 변경하는 일부 조치를 취하는지 여부를 지정합니다. 가정 메소드가 외부에 영향을 주지 않는 최적화는 EXTERNAL ACTION을 지정하여 방지합니다. 예를 들어, 메시지 전송, 벨 울림 또는 파일에 레코드 기록.

READS SQL DATA 또는 CONTAINS SQL

실행할 수 있는 SQL문의 유형을 나타냅니다. 지원되지 않는 SQL문은 RETURN문이므로, 표현식이 부속 조회인지 여부에 따라 구별을 수행해야 합니다.

READS SQL DATA

SQL 데이터를 수정하지 않는 SQL문을 메소드로 실행할 수 있음을 나타냅니다(SQLSTATE 42985). 별명은 SQL문에서 참조할 수 없습니다(SQLSTATE 42997).

CONTAINS SQL

SQL 데이터를 읽거나 수정하지 않는 SQL문을 메소드로 실행할 수 있음을 나타냅니다(SQLSTATE 42985).

CALLED ON NULL INPUT

이 선택적 절은 인수가 널(NULL)인지 여부에 관계없이 사용자 정의 메소드가 호출됨을 나타냅니다. 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 그러나 널(NULL) 인수 값의 테스트는 메소드에 종속적입니다.

계열 호환성을 위해 값 NULL CALL을 CALLED ON NULL INPUT의 동의어로서 사용할 수 있습니다.

external-routine-characteristics

LANGUAGE

이 필수 절은 사용자 정의 메소드 내용이 작성되는 언어 인터페이스 규약을 지정하는 데 사용됩니다.

C 이는 데이터베이스 관리 프로그램이 사용자 정의 메소드를 C 함수처럼 호출할 것임을 의미합니다. 사용자 정의 메소드는 표준 ANSI C 프로토타입으로 정의된 C 언어 호출 및 연계 변환에 따라야 합니다.

JAVA

이는 데이터베이스 관리 프로그램이 사용자 정의 메소드를 Java 클래스의 메소드처럼 호출할 것임을 의미합니다.

OLE

이는 데이터베이스 관리 프로그램이 사용자 정의 메소드를 OLE 자동 오브젝트로 노출되는 메소드처럼 호출할 것임을 의미합니다. 메소드는 OLE 자동 프로그래머 참조서에 설명된 대로 OLE 자동 데이터 유형 및 호출 메카니즘에 따라야 합니다.

LANGUAGE OLE는 Windows 32 비트 운영 체제에 저장된 사용자 정의 메소드에 대해서만 지원됩니다.

PARAMETER STYLE

이 절은 매개변수를 메소드로 전달하고 메소드로부터 값을 리턴하는 데 사용되는 규약 지정에 사용됩니다.

DB2SQL

OLE 자동 오브젝트에 의해 노출되는 메소드 또는 C 언어 호출 및 연계 규약에 따르는 외부 메소드로 매개변수를 전달하고 메소드로부터 값을 리턴하는 규약을 지정하는 데 사용됩니다. 이것은 LANGUAGE C 또는 LANGUAGE OLE를 사용할 때 지정해야 합니다.

DB2GENERAL

Java 클래스의 메소드로 정의된 외부 메소드로 매개변수를 전달하고 메소드로부터 값을 리턴하는 규약을 지정하는 데 사용됩니다. 이것은 LANGUAGE JAVA를 사용할 때만 지정할 수 있습니다.

값 DB2GENRL는 DB2GENERAL의 동의어로 사용할 수 있습니다.

*응용프로그램 개발 안내서*에 매개변수 전달에 대해 자세히 나와 있습니다.

DETERMINISTIC 또는 NOT DETERMINISTIC

이 선택적 절은 메소드가 항상 주어진 인수 값에 대해 같은 결과를 리턴하는지 여부(DETERMINISTIC) 또는 메소드가 결과에 영향을 주는 몇몇 상태 값에 따라 달라지는지 여부(NOT DETERMINISTIC)를 지정합니다. 즉, DETERMINISTIC 메소드는 항상 동일한 입력으로 계속되는 호출로부터같은 결과를 리턴해야 합니다. 동일한 입력이 항상 동일한 결과를 산출한다는 점을 이용하는 최적화는 NOT DETERMINISTIC를 지정하여 방지할 수 있습니다.

NOT DETERMINISTIC 메소드의 예로는 한 부서내 한 직원의 일련 번호를 랜덤으로 리턴하는 메소드를 들 수 있습니다. 다각형의 영역을 계산하는 메소드는 DETERMINISTIC 메소드의 예입니다.

FENCED 또는 NOT FENCED

이 절은 메소드가 데이터베이스 관리 프로그램 운영 환경의 프로세스 또는 주소 공간에서 실행하기에 "안전"여부(NOT FENCED) 또는 그렇지 않은지(FENCED)를 지정합니다.

메소드가 FENCED로 등록된 상태라면, 데이터베이스 관리 프로그램은 내부 자원(예: 데이터 버퍼)을 메소드에 의한 액세스로부터 보호합니다. 대부분의 메소드는 FENCED 또는 NOT FENCED로 실행하기 위한 옵션을 가집니다. 일반적으로, NOT FENCED로 수행 중인 메소드는 물론 FENCED로 수행 중인 메소드도 수행되지 않습니다.

주: 제대로 검사되지 않은 메소드의 NOT FENCED 사용은 DB2 무결성을 위태롭게 할 수 있습니다. DB2는 발생할 수 있는 부주의로 인한 실패의 많은 일반 유형에 대해 몇 가지 예방책을 취하나, NOT FENCED 사용자 정의 메소드가 정의된 경우에는 완벽한 무결성을 보증할 수 없습니다.

FENCED 사용은 NOT FENCED보다 더 우수한 데이터베이스 무결성 보호를 제공하는 반면, 제대로 코드화, 검토 및 테스트되지 않은 FENCED 메소드는 DB2 실패를 야기시킬 수도 있습니다.

대부분의 메소드는 FENCED 또는 NOT FENCED로 수행할 수 있어야 합니다. LANGUAGE OLE 메소드에 대해서는 FENCED만을 지정할 수 있습니다(SQLSTATE 42613).

FENCED 메소드의 경우, AS LOCATOR절을 지정할 수 없습니다 (SQLSTATE 42613).

FENCED에서 NOT FENCED로 변경하려면, 먼저 메소드를 삭제한 다음 다시 작성함으로써 메소드를 다시 등록해야 합니다.

메소드를 MOT FENCED로 등록하기 위해서는 SYSADM 권한, DBADM 권한 또는 특수 권한(CREATE_NOT_FENCED)이 필요합니다.

RETURNS NULL ON NULL INPUT 또는 CALLED ON NULL INPUT

이 선택적 절은 주제가 아닌 인수가 널(NULL)인 경우 외부 메소드 호출을 피하는 데 사용할 수 있습니다.

RETURNS NULL ON NULL INPUT이 지정된 경우와 실행시 메소드 인수 중 하나가 널(NULL)인 경우, 메소드는 호출되지 않고 결과는 널(NULL) 값입니다.

CALLED ON NULL INPUT이 지정되면, 널(NULL) 인수의 수에 관계없이 메소드가 호출됩니다. 널(NULL) 값을 리턴하거나 정상(널(NULL) 값이 아닌) 값을 리턴할 수 있습니다. 그러나 널(NULL) 인수 값의 테스트는 메소드에 종속적입니다.

백워드 및 계열 호환성을 위해 값 NULL CALL을 CALLED ON NULL INPUT의 동의어로서 사용할 수 있습니다. 마찬가지로, NOT NULL CALL이 RETURNS NULL ON NULL INPUT에 대한 동의어로서 사용될 수도 있습니다.

이 스펙이 무시되는 두 가지 경우가 있습니다.

- 주제 인수가 널(NULL)인 경우, 이 경우 메소드가 실행되지 않고 결과는 널(NULL)입니다.
- 메소드가 매개변수를 하나도 가지지 않도록 정의된 경우, 이 경우에는 이 널(NULL) 인수 조건이 발생할 수 없습니다.

NO SQL

이 필수 절은 메소드가 어느 SQL문도 발행할 수 없음을 나타냅니다. 발행하는 경우에는 실행시 오류가 발생합니다(SQLSTATE 38502).

EXTERNAL ACTION 또는 NO EXTERNAL ACTION

이 선택적 절은 메소드가 데이터베이스 관리 프로그램에서 관리하지 않는 오브젝트의 상태를 변경하는 일부 조치를 취하는지 여부를 지정합니다. 가정 메소드가 외부에 영향을 주지 않는 최적화는 EXTERNAL ACTION을 지정하여 방지합니다.

NO SCRATCHPAD 또는 SCRATCHPAD *길이*

이 선택적 절은 외부 메소드에 대해 스크래치 패드가 제공될 것인지 여부를 지정하는 데 사용할 수 있습니다. 메소드를 다시 입력하도록 강력하게 권장되므로, 스크래치 패드는 하나의 호출에서 다음 호출로 "상태를 저장" 하는 메소드 수단을 제공합니다.

SCRATCHPAD가 지정되면, 첫번째 사용자 정의 메소드 호출시 외부 메소드가 사용할 스크래치 패드용 메모리가 할당됩니다. 이 스크래치패드는 다음 특성을 갖습니다.

- 지정된 경우 *길이*는 스크래치 패드의 *크기*를 바이트 단위로 설정하며 이 *크기*는 1에서 32 767사이여야 합니다(SQLSTATE 42820). 기본값은 100입니다.
- 모두 X'00''로 초기화됩니다.
- 범위는 SQL문입니다. SQL문의 외부 메소드에 대한 참조당 하나의 스크래치 패드가 존재합니다.

따라서 다음 명령문의 메소드 X가 SCRATCHPAD 키워드로 지정되면 세 개의 스크래치 패드가 지정됩니다.

```
SELECT A, X..(A) FROM TABLEB
   WHERE X..(A) > 103 OR X..(A) < 19
```

ALLOW PARALLEL이 지정되거나 기본 설정되면, 범위가 위와 달라집니다. 메소드가 여러 파티션에서 실행되는 경우, SQL문내 메소드의 각 참조에 대해 해당 메소드가 처리되는 각 파티션에 하나의 스크래치 패드가

지정됩니다. 마찬가지로 파티션 내 병렬 처리를 사용하여 조회가 실행되면 네 개 이상의 스크래치 패드가 지정될 수 있습니다.

스크래치 패드는 영속적입니다. 그 내용은 하나의 외부 메소드 호출에서 다음 외부 메소드 호출로 유지됩니다. 하나의 호출에서 외부 메소드에 의한 스크래치 패드 변경은 다음 호출에서도 유지됩니다. 데이터베이스 관리 프로그램이 각 SQL문 실행의 시작시에 스크래치패드를 초기 설정합니다. 데이터베이스 관리 프로그램이 각 부속 조회 실행의 시작시에 스크래치패드를 재설정합니다. FINAL CALL 옵션이 지정되면, 시스템에서 스크래치 패드를 재설정하기 전에 최종 호출을 발행합니다.

스크래치 패드는 외부 메소드가 획득할 수 있는 시스템 자원(예: 메모리)의 중앙 지점으로서 사용될 수 있습니다. 메소드는 첫번째 호출시 메모리를 획득하고 그 주소를 스크래치 패드에 보존하며 후속 호출시 이를 참조할 수 있습니다.

시스템 자원이 획득되는 경우 FINAL CALL 키워드도 지정해야 합니다. 그러면 외부 메소드가 획득된 모든 시스템 자원을 해제할 수 있도록 명령문의 끝에서 특수 호출이 작성됩니다.

SCRATCHPAD가 지정되면, 각 사용자 정의 메소드 호출시 추가 인수가 스크래치 패드를 주소지정하는 외부 메소드로 전달됩니다.

NO SCRATCHPAD가 지정되면, 스크래치 패드가 외부 메소드에 할당되거나 전달되지 않습니다.

NO FINAL CALL 또는 FINAL CALL

이 선택적 절은 외부 메소드에 대한 최종 호출이 이루어질지 여부를 지정합니다. 그러한 최종 호출의 목적은 외부 메소드가 획득한 시스템 자원을 해제할 수 있게 하는 것입니다. 이는 외부 메소드가 메모리와 같은 시스템 자원을 획득하고 이를 스크래치 패드에 저장하는 상황에서 SCRATCHPAD 키워드와 함께 유용할 수 있습니다.

FINAL CALL이 지정되면, 실행시 추가 인수가 호출 유형을 지정하는 외부 메소드로 전달됩니다. 호출 유형은 다음과 같습니다.

- 정상 호출: SQL 인수가 전달되고 결과 리턴이 예상됩니다.

- 첫번째 호출: 이러한 특정 SQL문에서 메소드의 특정 참조에 대한 외부 메소드의 첫번째 호출. 첫번째 호출은 정상적인 호출입니다.
- 최종 호출: 메소드가 자원을 해제할 수 있도록 해주는 외부 메소드에 대한 최종 호출. 마지막 호출은 정상적인 호출이 아닙니다. 이 마지막 호출은 다음 상황에서 발생합니다.
  - 명령문의 끝: 이러한 경우는 커서 지향 명령문에 대한 커서가 닫히거나 명령문이 실행될 때 발생합니다.
  - 트랜잭션의 끝: 이 경우는 정상적인 명령문 끝이 발생하지 않을 때 발생됩니다. 예를 들어, 응용프로그램의 논리는 어떤 이유로 커서 닫기를 생략할 수 있습니다.

  WITH HOLD로 정의된 커서가 열려 있는 동안 확약 조작이 발생하면, 그 다음의 커서를 닫을 때 또는 응용프로그램 종료시 최종 호출이 수행됩니다.

NO FINAL CALL이 지정되면, "호출 유형" 인수가 외부 메소드로 전달되지 않고 최종 호출이 이루어지지 않습니다.

ALLOW PARALLEL 또는 DISALLOW PARALLEL

이 선택적 절은 메소드에 대한 단일 참조의 경우 메소드 호출이 병렬 처리될 수 있는지 여부를 지정합니다. 일반적으로 대부분의 메소드 호출을 병렬 처리 가능해야 하나, 병렬 처리할 수 없는 메소드(스크래치 패드의 단일 사본에 종속적인 메소드)가 있을 수 있습니다. 메소드에 대해 ALLOW PARALLEL 또는 DISALLOW PARALLEL이 지정된 경우 DB2는 이 스펙을 승인합니다.

메소드에 적절한 키워드 판별시에는 다음 질문을 고려해야 합니다.

- 모든 메소드 호출이 완벽하게 서로 독립적인가? 그럴 경우, ALLOW PARALLEL을 지정하십시오.
- 각 메소드 호출시 다음 호출과 관련된 값을 제공하여 스크래치 패드를 갱신하는가(예: 카운터 증가)? 그럴 경우 DISALLOW PARALLEL을 지정하거나 기본값을 사용하십시오.

- 하나의 파티션에 대해서만 발생해야 하는 메소드로 수행되는 외부 조치가 있는가? 그럴 경우 DISALLOW PARALLEL을 지정하거나 기본값을 사용하십시오.
- 비용이 많이 드는 초기설정 처리를 최소한의 횟수로 수행할 수 있도록 스크래치패드가 사용되었는가? 그럴 경우, ALLOW PARALLEL을 지정하십시오.

경우에 따라 모든 외부 메소드 내용이 데이터베이스의 모든 파티션에서 사용 가능한 디렉토리에 있어야 합니다.

구문 도표는 기본값이 ALLOW PARALLEL임을 나타냅니다. 단, 다음 옵션 중 하나 이상이 명령문에 지정되어 있을 경우 기본값은 DISALLOW PARALLEL입니다.

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

NO DBINFO 또는 DBINFO

이 선택적 절은 DB2에서 제공하는 고유한 특정 정보가 추가 호출시 인수로 메소드에 전달되는지(DBINFO) 또는 전달되지 않는지(NO DBINFO)를 지정합니다. NO DBINFO가 기본값입니다. DBINFO는 LANGUAGE OLE에 대해 지원하지 않습니다(SQLSTATE 42613).

DBINFO가 지정된 경우, 다음 정보를 포함하는 메소드로 구조가 전달됩니다.

- 데이터베이스 이름 - 현재 연결된 데이터베이스의 이름.
- 응용프로그램 ID - 데이터베이스로의 각 연결에 형성되는 고유한 응용프로그램 ID.
- 응용프로그램 권한 부여 ID - 이 메소드와 응용프로그램간의 중첩된 메소드에 관계없이 응용프로그램 권한 부여 ID.
- 코드 페이지 - 데이터베이스 코드 페이지를 식별합니다.
- 스키마 이름 - 테이블 이름과 완전히 같은 조건하에서 스키마의 이름을 포함합니다. 그렇지 않으면 공백입니다.

- 테이블 이름 - 메소드 참조가 UPDATE문의 SET절 오른쪽이거나 INSERT문의 VALUES 목록에 있는 한 항목인 경우, 갱신 또는 삽입 중인 테이블의 규정화되지 않은 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 컬럼 이름 - 테이블 이름과 완전히 같은 조건하에서, 갱신되거나 삽입 중인 컬럼의 이름을 포함합니다. 그렇지 않으면 공백입니다.
- 데이터베이스 버전/릴리스 - 메소드를 호출하는 데이터베이스 서버의 버전, 릴리스 및 수정 레벨을 식별합니다.
- 플랫폼 - 서버의 플랫폼 유형을 포함합니다.
- 테이블 메소드 결과 컬럼 수 - 메소드에 적용 불가능합니다.

*응용프로그램 개발 안내서*에 구조에 대한 세부사항 정보 및 메소드로 전달 방법이 자세히 나와 있습니다.

주

- 아직 존재하지 않는 스키마 이름을 사용하여 구조화 유형을 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 속성 없이 정의된 구조화 서브유형은 수퍼 유형으로부터 모든 속성을 상속하는 서브유형을 정의합니다. UNDER절이나 다른 속성이 지정되지 않았다면 이 유형은 속성없는 유형 계층의 루트 유형입니다.
- 유형 계층에 새 부속 유형을 추가하면 패키지가 무효화될 수 있습니다. 패키지가 새 유형의 상위 유형에 종속적인 경우 이 패키지를 무효화할 수 있습니다. 그러한 종속성은 TYPE 술어 또는 TREAT 스펙의 사용 결과입니다.
- 구조화 유형은 4082개 이하의 속성을 가질 수 있습니다(SQLSTATE 54050).
- 함수와 같은 시그니처를 가지는 메소드 스펙은 허용되지 않습니다(메소드의 주제 유형과 함수의 첫번째 매개변수 유형 비교).
- 주제 유형 T를 갖는 메소드 MT는 다음 사항 모두가 만족되는 경우 주제 유형 S인 또 다른 메소드 MS를 대체하도록 정의되었습니다.

- MT 및 MS가 같은 규정화되지 않은 이름과 같은 수의 매개변수를 가집니다.
- T가 S의 적절한 부속 유형입니다.
- MT의 주제가 아닌 매개변수 유형이 해당되는 MS의 주제가 아닌 매개변수 유형과 같습니다. 길이 및 정밀도에 관계없이 VARCHAR과 같은 기본 유형에도 "동일하게" 적용됩니다.

다른 메소드를 대체할 수 있는 메소드가 없거나 다른 메소드에 의해 대체될 수 있는 메소드가 없을 수 있습니다(SQLSTATE 42745). 더우기 함수와 메소드가 대체 관계에 있지 않을 수도 있습니다. 이는 함수가 주제 S를 첫번째 매개변수로 갖는 메소드인 경우, 이 함수가 S에 대한 상위 유형의 다른 메소드를 대체해서는 안되고 S에 대한 부속 유형의 다른 메소드에 의해 대체되어서도 안됨을 의미합니다.

- 구조화 유형을 작성하면 해당 유형과 함께 사용될 함수 및 메소드 집합이 자동으로 생성됩니다. 모든 함수와 메소드는 구조화 유형과 같은 스키마에 생성됩니다. 생성된 함수 또는 메소드의 시그니처가 이 스키마에 있는 기존 함수의 시그니처와 상충되거나 이 시그니처를 대체하는 경우, 명령문은 실패합니다(SQLSTATE 42710). 생성된 함수 또는 메소드는 구조화 유형을 삭제하지 않고 삭제할 수 없습니다(SQLSTATE 42917). 다음 함수와 메소드가 생성됩니다.
  - 함수
    - 참조 비교

      이름이 =, <>, <, <=, >, >=인 여섯 가지 비교 함수가 참조 유형 REF(*type-name*)에 대해 생성됩니다. 이들 각 함수는 유형 REF(*type-name*)의 두 매개변수를 취하고, 참, 거짓 또는 알 수 없음을 리턴합니다. REF(*type-name*)에 대한 비교 연산자는 REF(*type-name*)의 기본 데이터 유형에 대한 비교 연산자와 같은 동작을 하도록 정의되어 있습니다.[85]

      비교시 참조 유형의 범위는 고려되지 않습니다.

85. 유형 계층의 모든 참조는 동일한 참조 표시 유형을 가집니다. 이것은 S와 T가 공동의 상위 유형을 가질 경우, REF(S)와 REF(T)가 비교 가능하도록 합니다. 0ID 컬럼의 고유성은 하나의 테이블 계층내에서만 강요되므로, 한 테이블 계층의 REF(T) 값이 다른 테이블 계층의 REF(T) 값과 "동일"할 수 있으며 이는 다른 행을 참조하더라도 마찬가지입니다.

- 유형변환 함수

  생성된 참조 유형 REF(*type-name*)와 이 참조 유형의 기본 데이터 유형 사이의 유형변환을 위해 두 개의 유형변환 함수가 생성되었습니다.

  - 기본 유형에서 참조 유형으로의 유형변환을 위한 함수 이름은 내재된 또는 명시적 *funcname1*입니다.

    이 함수의 형식은 다음과 같습니다.

    ```
    CREATE FUNCTION funcname1 (rep-type)
       RETURNS REF(type-name) ...
    ```

  - 참조 유형에서 기본 유형으로의 유형변환을 위한 함수 이름은 내재된 또는 명시적 *funcname2*입니다.

    이 함수의 형식은 다음과 같습니다.

    ```
    CREATE FUNCTION funcname2 ( REF(type-name) )
      RETURNS rep-type ...
    ```

  일부 rep-type의 경우, 상수로부터의 유형변환 처리를 위해 *funcname1*로 생성된 유형변환 함수가 추가되었습니다.

  - *rep-type*이 SMALLINT인 경우, 추가로 생성된 유형변환 함수는 다음 형식을 가집니다.

    ```
    CREATE FUNCTION funcname1 (INTEGER)
       RETURNS REF(type-name)
    ```

  - *rep-type*이 CHAR(n)인 경우, 추가로 생성된 유형변환 함수는 다음 형식을 가집니다.

    ```
    CREATE FUNCTION funcname1 ( VARCHAR(n))
       RETURNS REF(type-name)
    ```

  - *rep-type*이 GRAPHIC(n)인 경우, 추가로 생성된 유형변환 함수는 다음 형식을 가집니다.

    ```
    CREATE FUNCTION funcname1 (VARGRAPHIC(n))
       RETURNS REF(type-name)
    ```

  SQL문의 이러한 연산자와 유형변환 함수의 정상적인 사용을 위해 구조화 유형의 스키마 이름이 SQL 경로에 포함되어야 합니다(*응용프로그램 개발 안내서*에 설명된 바와같이 1174 페이지의 『SET PATH』 또는 FUNCPATH BIND 옵션을 참조하십시오).

- 구성자 함수

  구성자 함수는 구성될 유형의 새로운 인스턴스를 허용하기 위해 생성됩니다. 이러한 새 인스턴스는 상위 유형으로부터 계승된 속성을 포함하여 해당 유형의 모든 속성에 대해 널(NULL)을 가지게 됩니다.

  다음은 생성된 구성자 함수의 형식입니다.

```
CREATE FUNCTION type-name ( )
   RETURNS type-name
  ...
```

  NOT INSTANTIABLE이 지정된 경우에는 구성자 함수가 생성되지 않습니다. 구조화 유형이 DATALINK 유형인 속성을 가지는 경우, 구성자 함수의 호출은 실패합니다(SQLSTATE 428ED).

– 메소드

- 관찰자(Observer) 메소드

  구조화 유형의 속성 각각에 대한 관찰자 메소드가 정의됩니다. 관찰자 메소드는 각 속성에 대해 해당 속성의 유형을 리턴합니다. 주제가 널(NULL)인 경우, 관찰자 메소드는 해당 속성 유형의 널(NULL) 값을 리턴합니다.

  예를 들어, 구조화 유형 ADDRESS의 인스턴스 속성은 `C1..STREET`, `C1..CITY`, `C1..COUNTRY` 및 `C1..CODE`를 사용하여 관찰할 수 있습니다.

  생성된 관찰자 메소드의 메소드 시그니처는 다음 명령문이 실행된 경우와 같습니다.

```
CREATE TYPE  type-name
      ...
   METHOD attribute-name()
      RETURNS attribute-type
```

  여기서, *type-name*은 구조화된 유형 이름입니다.

- 변화(Mutator) 메소드

  구조화 유형의 속성 각각에 대한 유형 보존 변화 메소드가 정의됩니다. 변화 메소드를 사용하여 구조화 유형의 인스턴스내 속성을 변경하십시오. 변화 메소드는 각 속성에 대해 사본의 명명된 속성에 인수를 지정함으로써 수정된 주제의 사본을 리턴합니다.

예를 들어, 구조화 유형 ADDRESS의 인스턴스는 C1..CODE('M3C1H7') 를 사용하여 변화시킬 수 있습니다. 주제가 널(NULL)인 경우, 변화 메소드는 오류를 일으킵니다(SQLSTATE 2202D).

생성된 변화 메소드의 메소드 시그니처는 다음 명령문이 실행된 경우와 같습니다.

```
CREATE TYPE type-name
    ...
  METHOD attribute-name (attribute-type)
    RETURNS type-name
```

속성 데이터 유형이 SMALLINT, REAL, CHAR 또는 GRAPHIC인 경우, 상수를 사용한 변화를 지원하기 위해 추가 변화 메소드가 생성됩니다.

- *attribute-type*이 SMALLINT인 경우, 추가 변화 메소드는 INTEGER 유형의 인수를 지원합니다.
- *attribute-type*이 REAL인 경우, 추가 변화 메소드는 DOUBLE 유형의 인수를 지원합니다.
- *attribute-type*이 CHAR인 경우, 추가 변화 메소드는 VARCHAR 유형의 인수를 지원합니다.
- *attribute-type*이 GRAPHIC인 경우, 추가 변화 메소드는 VARGRAPHIC 유형의 인수를 지원합니다.

- 구조화 유형이 컬럼 유형으로 사용되는 경우, 해당 유형의 인스턴스 길이는 수행시 1GB를 넘을 수 없습니다(SQLSTATE 54049).

- 기존의 구조화 유형에 대한 새 부속 유형을 작성하는 경우(컬럼 유형으로 사용하기 위해), 기존의 관련 구조화 유형의 지원으로 이미 작성된 변환 함수를 다시 검사하고 필요하다면 갱신해야 합니다. 새 유형이 주어진 유형과 같은 계층에 있는지 또는 중첩된 유형의 계층에 있는지에 따라, 이 유형과 연관된 기존의 변환 함수가 새 부속 유형에 의해 도입되는 새 속성 중 일부 또는 모두를 포함하도록 수정해야 합니다. 일반적으로 UDF 및 클라이언트 응용프로그램이 구조화 유형에 액세스할 수 있게 해주는 주어진 유형(또는 유형 계층)과 연관된 변환 함수 집합이므로, 주어진 복합 계층에서 모든 속성을 지원하는 변환 함수를 작성해야 합니다(즉, 모든 부속 유형 및 중첩된 구조화 유형의 임시 닫기 포함).

예

예 1: 부서에 대한 유형을 작성합니다.

```
CREATE TYPE DEPT AS
   (DEPT NAME     VARCHAR(20),
      MAX_EMPS INT)
      REF USING INT
   MODE DB2SQL
```

예 2: 관리 프로그램에 대한 부속 유형과 사원에 대한 유형으로 구성된 유형 계층을 작성합니다.

```
CREATE TYPE EMP AS
     (NAME      VARCHAR(32),
   SERIALNUM INT,
   DEPT      REF(DEPT),
      SALARY    DECIMAL(10,2) )
   MODE DB2SQL
CREATE TYPE MGR UNDER EMP AS
    (BONUS     DECIMAL(10,2))
            MODE DB2SQL
```

예 3: 주소에 대한 유형 계층을 작성합니다. 주소는 컬럼의 유형으로 사용됩니다. 인라인 길이가 지정되지 않으므로, DB2가 기본 길이를 계산합니다. 주소 유형 정의내에서 이 주소가 주어진 입력 주소가 되도록 하는 방법을 계산하는 외부 메소드를 캡슐화하십시오. CREATE METHOD문을 사용하여 메소드 내용을 작성하십시오.

```
CREATE TYPE address_t AS
   (STREET     VARCHAR(30),
    NUMBER     CHAR(15),
    CITY       VARCHAR(30),
    STATE      VARCHAR(10))
    NOT FINAL
    MODE DB2SQL
       METHOD SAMEZIP (addr address_t)
       RETURNS INTEGER
       LANGUAGE SQL
       DETERMINISTIC
       CONTAINS SQL
       NO EXTERNAL ACTION
       METHOD DISTANCE (address_t)
       RETURNS FLOAT
       LANGUAGE C
       DETERMINISTIC
```

```
        PARAMETER STYLE DB2SQL
        NO SQL
        NO EXTERNAL ACTION
CREATE TYPE germany_addr_t UNDER address_t AS
   (FAMILY_NAME VARCHAR(30))
    NOT FINAL
    MODE DB2SQL
CREATE TYPE us_addr_t UNDER address_t AS
    (ZIP VARCHAR(10))
     NOT FINAL
     MODE DB2SQL
```

예 4: 중첩된 구조화 유형 속성을 갖는 유형을 작성하십시오.

```
CREATE TYPE PROJECT AS
      (PROJ_NAME  VARCHAR(20),
       PROJ_ID    INTEGER,
       PROJ_MGR   MGR,
       PROJ_LEAD  EMP,
       LOCATION   ADDR_T,
       AVAIL_DATE DATE)
       MODE DB2SQL
```

CREATE TYPE MAPPING

CREATE TYPE MAPPING문은 다음 데이터 유형들간의 맵핑을 작성합니다.

- 연합 데이터베이스에 정의될 데이터 소스 테이블이나 뷰의 컬럼의 데이터 유형
- 연합 데이터베이스에 이미 정의되어 있는 해당 데이터 유형.

맵핑은 연합 데이터베이스 데이터 유형을 (1) 지정된 데이터 소스나 (2) 지정된 데이터 소스의 범위(예를 들어, 특정 유형 및 버전의 모든 데이터 소스)에 있는 데이터 유형과 연관시킬 수 있습니다.

기존 것이 적당한 경우에만 데이터 유형 맵핑이 작성되어야 합니다.

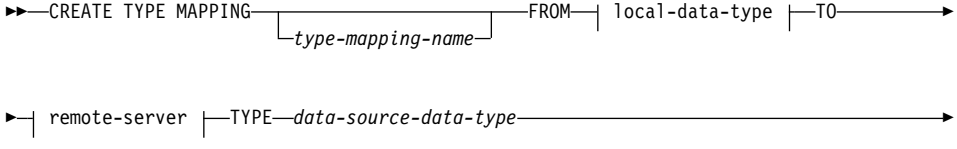
호출

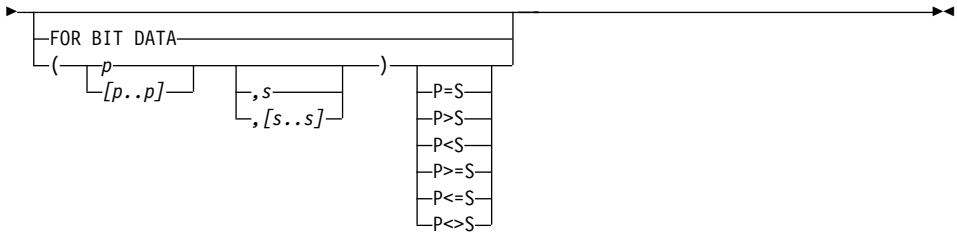
이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

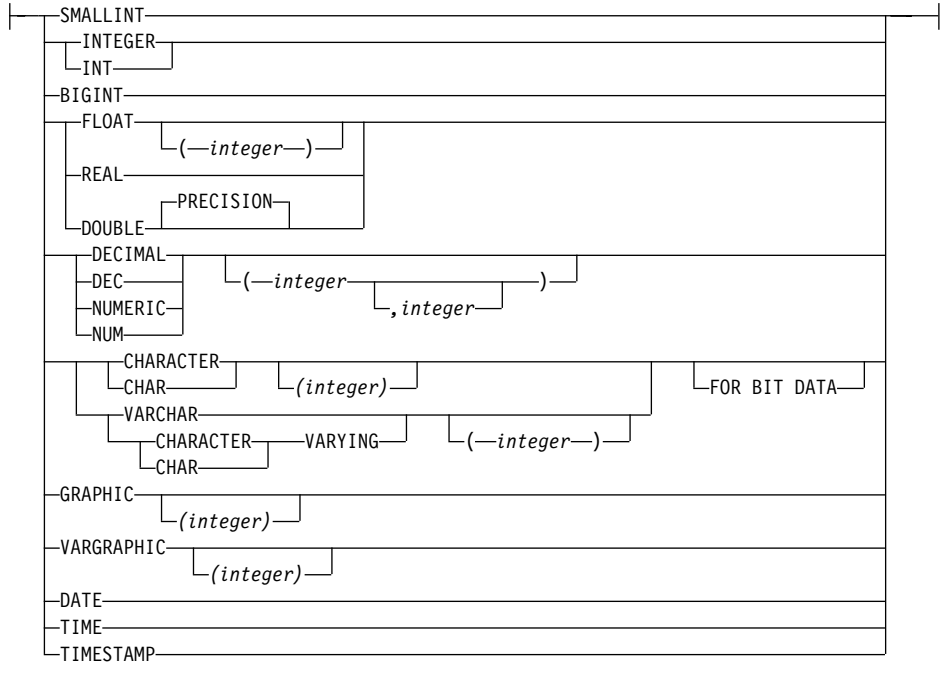
명령문의 권한 부여 ID에 의해 보유된 특권은 SYSADM 또는 DBADM 권한을 가져야 합니다.

구문

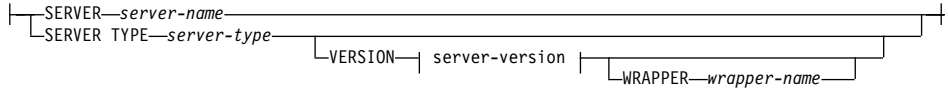




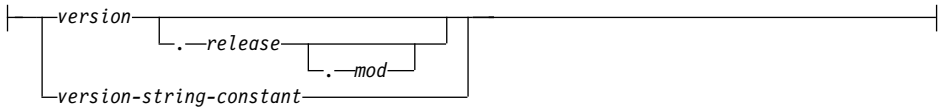
local-data-type:



remote-server:



server-version:



설명

type-mapping-name

데이터 유형 맵핑의 이름을 지정합니다. 이 이름은 카탈로그에 이미 기술되어 있는 데이터 유형 맵핑을 식별해선 안 됩니다. *type-mapping-name*이 지정되지 않으면 고유한 이름이 생성됩니다.

local-data-type

연합 데이터베이스에 정의되는 데이터 유형을 식별합니다. 스키마 이름 없이 *local-data-type*이 지정되면, 유형 이름은 SQL 경로(정적 SQL인 경우에는 FUNCPATH 사전 처리 옵션, 동적 SQL인 경우에는 CURRENT PATH 레지스터에 의해 정의)에서 스키마를 검색하여 분석됩니다. *local-data-type*에 대한 길이 또는 정밀도(및 스케일)이 지정되지 않은 경우, 값은 *source-data-type*에서 결정됩니다.

*local-data-type*은 LONG VARCHAR, LONG VARGRAPHIC, DATALINK, 대형 오브젝트(LOB) 유형 또는 사용자 정의 유형일 수 없습니다(SQLSTATE 42806).

SERVER *server-name*

*data-source-data-type*이 정의되는 데이터 소스의 이름을 지정합니다.

SERVER TYPE *server-type*

*data-source-data-type*이 정의되는 데이터 소스의 유형을 식별합니다.

VERSION

*data-source-data-type*이 정의되는 데이터 소스의 버전을 식별합니다.

version

버전 번호를 지정합니다. *version*은 정수여야 합니다.

release

*version*으로 표시된 버전의 릴리스 번호를 지정합니다. *release*는 정수여야 합니다.

mod

*release*로 표시된 릴리스의 수정 번호를 지정합니다. *mod*는 정수여야 합니다.

version-string-constant

완전한 버전 지정을 합니다. *version-string-constant*는 단일 값(예: '8i') 이거나 *version*, *release* 및 *mod*의 결합된 값(예: '8.0.3')일 수 있습니다.

WRAPPER *wrapper-name*

server-type 및 *server-version*에 의해 표시된 유형 및 버전의 데이터 소스와 상호 작용하기 위해 연합 서버가 사용하는 랩퍼의 이름을 지정합니다.

TYPE *data-source-data-type*

*local-data-type*에 맵핑되고 있는 데이터 소스 데이터 유형을 지정합니다. *data-source-data-type*이 데이터 소스에서 스키마에 의해 규정화되지 않으면, 이 규정자를 지정하는 것이 허용됩니다.

*data-source-data-type*은 내장 데이터 유형이어야 합니다. 사용자 정의 유형은 허용되지 않습니다. 유형에 축약 형태와 긴 형태(CHAR 및 CHARACTER)가 있으면, 축약 형태가 지정되어야 합니다.

p 십진 데이터 유형의 경우, *p*는 값이 가질 수 있는 최대 자리 수를 지정합니다. 문자 데이터에 대한 다른 모든 데이터 유형의 경우, *p*는 값이 가질 수 있는 최대 문자 수를 지정합니다. *p*는 데이터 유형에 대해 유효해야 합니다 (SQLSTATE 42611). *p*가 지정되지 않고 데이터 유형이 이를 필요로 하면, 시스템은 가장 잘 일치하는 것을 결정할 것입니다.

[p..p]

십진 데이터 유형의 경우, *[p..p]*는 값이 가질 수 있는 최대 자리 수를 지정합니다. 문자 데이터에 대한 다른 모든 데이터 유형의 경우, *[p..p]*는 값이 가질 수 있는 최대 문자 수를 지정합니다. 모든 경우에 있어, 최대값은 최소값과 같거나 커야 합니다. 그리고 두 숫자 모두 데이터 유형에 대해 유효해야 합니다 (SQLSTATE 42611).

s 십진 데이터 유형의 경우, *s*는 십진 분리점 오른쪽으로 허용 가능한 최대 자리

수를 지정합니다. 이 숫자는 데이터 유형에 대해 유효해야 합니다(SQLSTATE 42611). 숫자가 지정되지 않고, 데이터 유형이 이를 필요로 하면, 시스템은 가장 잘 일치하는 것을 결정할 것입니다.

[s..s]

십진 데이터 유형의 경우, *[s..s]*는 십진 분리점 오른쪽으로 허용 가능한 최대 자리수를 지정합니다. 최대값은 최소값과 같거나 커야 합니다. 그리고 두 숫자 모두 데이터 유형에 대해 유효해야 합니다(SQLSTATE 42611).

P [operand] **S**

십진 데이터 유형의 경우, P [operand] S는 최대 허용 가능 정밀도와 십진 분리점 오른쪽으로 허용 가능한 최대 자리수와의 비교를 지정합니다. 예를 들어, 피연산자 =는 허용 가능 정밀도와 십진 수에서 허용되는 최대 자리수가 같음을 나타냅니다. 강화하는 점검 레벨이 필요한 경우에만 P [operand] S를 지정하십시오.

FOR BIT DATA

*data-source-data-type*이 비트 데이터에 대한 것인지의 여부를 나타냅니다. 데이터 소스 유형 컬럼이 2진 값을 포함하는 경우에 이들 키워드가 필요합니다. 문자 데이터 유형에서 이것이 지정되지 않으면, 데이터베이스 관리 프로그램이 이 속성을 결정합니다.

주

주어진 작업 단위(UOW) 내의 CREATE TYPE MAPPING문은 다음 조건에서 처리될 수 없습니다.

- 명령문이 단일 데이터 소스를 참조하며, UOW가 이미 이 데이터 소스 내의 테이블이나 뷰에 대한 별명을 참조하는 SELECT문을 포함합니다.
- 명령문이 데이터 소스의 카테고리를 참조하며(예를 들어, 특정 유형 및 버전의 모든 데이터 소스), UOW가 이미 이들 데이터 소스 중 하나 안에 있는 테이블이나 뷰에 대한 별명을 참조하는 SELECT문을 포함합니다.

예

예 1: SYSIBM.DATE와 Oracle 데이터 소스에 있는 Oracle 데이터 유형 DATE 간의 맵핑을 작성하십시오.

```
CREATE TYPE MAPPING MY_ORACLE_DATE
   FROM SYSIBM.DATE
   TO SERVER TYPE ORACLE
   TYPE DATE
```

예 2: SYSIBM.DECIMAL(10,2)과 데이터 소스 ORACLE1에 있는 Oracle 데이터 유형 NUMBER([10..38],2)간의 맵핑을 작성하십시오.

```
CREATE TYPE MAPPING MY_ORACLE_DEC
   FROM SYSIBM.DECIMAL(10,2)
   TO SERVER ORACLE1
   TYPE NUMBER([10..38],2)
```

CREATE USER MAPPING

CREATE USER MAPPING문은 연합 데이터베이스를 사용하는 권한 부여 ID와 지정된 데이터 소스에서 사용할 권한 부여 ID 및 암호간의 맵핑을 정의합니다.

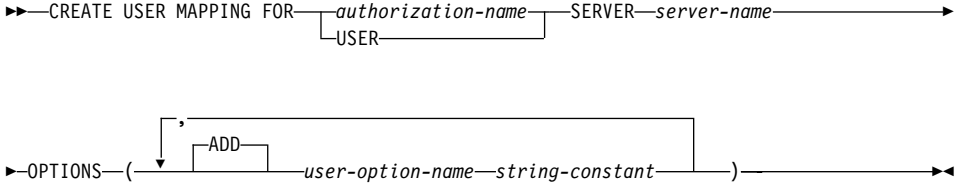
호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID가 데이터 소스에 맵핑되는 권한 부여 이름과 다르면, 권한 부여 ID가 SYSADM 또는 DBADM 권한을 포함해야 합니다. 그렇지 않으면, 권한 부여 ID 및 권한 부여 이름이 일치하는 경우 어떠한 특권도 필요하지 않습니다.

구문



설명

authorization-name

사용자나 응용프로그램이 연합 데이터베이스에 연결하는 권한 부여 이름을 지정합니다. 이 이름은 *server-name*에 의해 표시된 데이터 소스가 액세스될 수 있는 식별자로 맵핑될 것입니다.

USER

특수 레지스터 USER에 있는 값. USER가 지정될 때에는, CREATE USER

MAPPING문의 권한 부여 ID가 REMOTE_AUTHID 사용자 옵션에서 지정되는 데이터 소스 권한 부여 ID로 맵핑됩니다.

SERVER *server-name*

맵핑 권한 부여 ID하에서 액세스 가능한 데이터 소스를 식별합니다.

OPTIONS

작동 가능화될 사용자 옵션을 나타냅니다. *user-option-name*의 설명과 설정값은 1412 페이지의 『사용자 옵션』에서 참조하십시오.

ADD

하나 이상의 사용자 옵션을 작동 가능화시킵니다.

user-option-name

작성되고 있는 사용자 맵핑을 완료하기 위해 사용될 사용자 옵션의 이름을 지정합니다.

string-constant

*user-option-name*에 대한 설정값을 문자열 상수로서 지정합니다.

주

- UOW가 이미 맵핑에 포함될 데이터 소스에서 테이블이나 뷰에 대한 별명을 참조하는 SELECT문을 포함하는 경우에는 주어진 작업 단위(UOW)에서 사용자 맵핑이 변경될 수 없습니다.

예

예 1: S1이라는 데이터 소스에 액세스하기 위해, 지역 데이터베이스에 대한 권한 부여 이름과 암호를 S1에 대한 사용자 ID와 암호로 맵핑해야 합니다. 권한 부여 이름은 RSPALTEN이고, 사용자 S1에 사용하는 ID 및 암호는 각가 SYSTEM과 MANAGER입니다.

```
CREATE USER MAPPING FOR RSPALTEN
   SERVER S1
   OPTIONS
   ( REMOTE_AUTHID 'SYSTEM',
      REMOTE_PASSWORD 'MANAGER' )
```

예 2: Marc가 DB2 데이터 소스에 액세스합니다. 그는 이제 특정 B2와 Oracle 테이블간에 조인 을 작성할 수 있도록 Oracle 데이터 소스에 대한 액세스를 필요

로 합니다. 그는 Oracle 데이터 소스에 대한 사용자 이름 및 암호를 확보합니다. 사용자 이름은 연합 데이터베이스에 대한 권한 부여 ID와 동일하나, Oracle과 연합 서버의 암호는 서로 다릅니다. 연합 데이터베이스에서 Oracle을 액세스하기 위해서는, 두 개의 암호를 서로 맵핑해야 합니다.

```
CREATE USER MAPPING FOR MARCR
   SERVER ORACLE1
   OPTIONS
   ( REMOTE_PASSWORD 'NZXCZY' )
```

CREATE VIEW

CREATE VIEW문은 하나 이상의 테이블, 뷰 또는 별명에 대한 뷰를 작성합니다.

호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 각 테이블에 대해 fullselect에서 식별되는 뷰 또는 별명:
  - 해당 테이블이나 뷰에서의 CONTROL 특권
  - 해당 테이블이나 뷰에서의 SELECT 특권

  다음 중 최소한 하나는 포함해야 합니다.
  - 뷰의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 뷰의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권

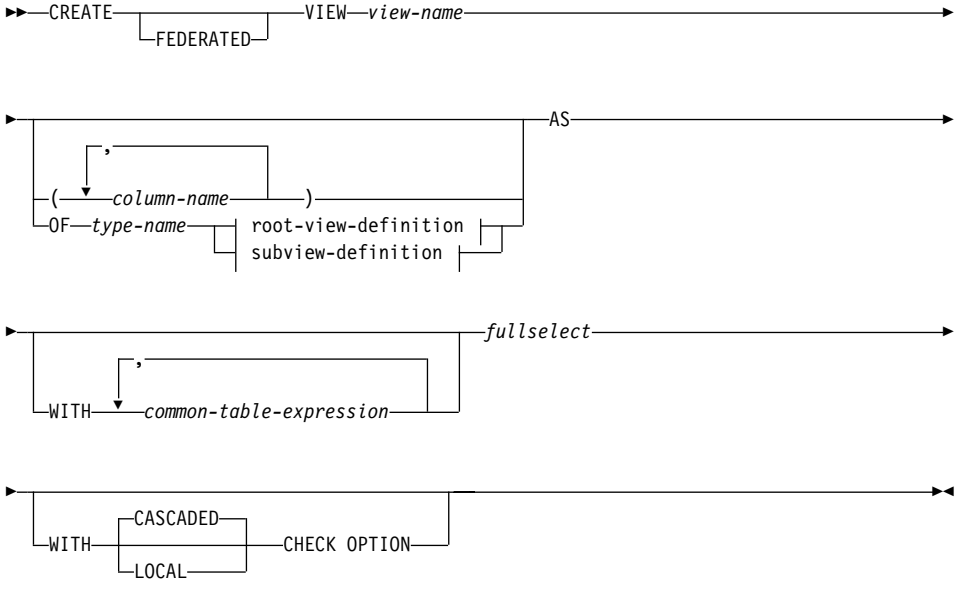
  부속 뷰를 작성하는 경우, 명령문의 권한 부여 ID는 다음과 같아야 합니다.
  - 테이블 계층의 루트 테이블 정의자와 같아야 합니다.
  - 부속 뷰의 기본 테이블에 SELECT WITH GRANT 권한이 있어야 합니다. 그렇지 않은 경우, 상위 뷰는 뷰 정의자가 아닌 다른 사용자에게 SELECT 특권을 권한 부여할 수 없습니다.

CREATE VIEW문에 지정된 테이블이나 뷰에 대해서는 그룹 특권이 고려되지 않습니다.

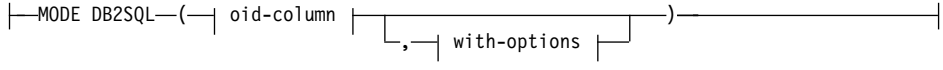
연합 데이터베이스 별명에서 뷰를 정의할 때에는 특권이 고려되지 않습니다. 별명으로 참조된 테이블이나 뷰에 대한 데이터 소스의 권한 부여 요건은 조회가 처리될 때 적용됩니다. 명령문의 권한 부여 ID는 서로 다른 원격 권한 부여 ID로 맵핑될 수도 있습니다.

뷰(view) 정의자가 SYSADM 권한이 있어 뷰만을 작성할 수 있는 경우, 정의자는 뷰를 작성할 목적으로 명시적인 DBADM 권한을 권한 부여받게 됩니다.

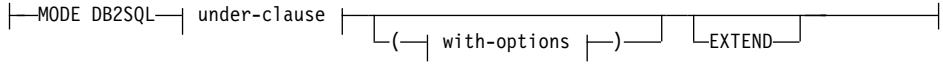
구문



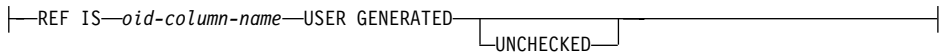
root-view-definition:



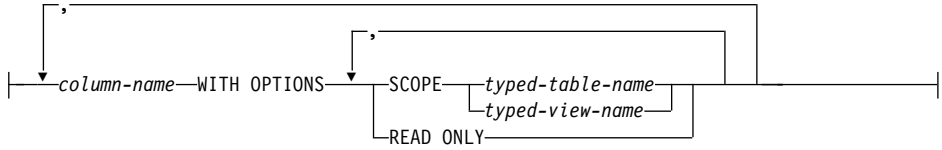
subview-definition:



oid-column:



with-options:



under-clause:

주: *common-table-expression* 및 *fullselect*의 구문에 대해서는 433 페이지의 『제 5장 조회』에서 참조하십시오.

설명

FEDERATED

작성 중인 뷰가 별명이나 OLEDB 테이블 함수를 참조함을 나타냅니다. OLEDB 테이블 함수나 별명이 직접 또는 간접적으로 fullselect내에서 참조되고 FEDERATED 키워드가 지정되지 않은 경우, CREATE VIEW문이 제출되면 경고가 발행됩니다(SQLSTATE 01639). 그러나 아직 뷰가 작성됩니다.

반대로 OLEDB 테이블 함수나 별명이 직접 또는 간접적으로 fullselect내에서 참조되지 않고 FEDERATED 키워드가 지정되는 경우, CREATE VIEW 문이 제출되면 오류가 발행됩니다(SQLSTATE 429BA). 뷰는 작성되지 않습니다.

view-name

뷰를 명명합니다. 암시적 또는 명시적인 규정자를 포함하여, 이름은 카탈로그에 설명된 테이블, 뷰, 별명(nickname) 또는 별명(alias)을 식별해서는 안 됩니다 규정자는 SYSIBM, SYSCAT, SYSFUN 또는 SYSSTAT(SQLSTATE 42939)이 될 수 없습니다.

이름은 실행 불능(inoperative) 뷰의 이름과 같을 수 있습니다(942 페이지의 『작동 불능 뷰』 참조). 이 경우, CREATE VIEW문에 지정된 새로운 뷰는 실행 불능 뷰를 대체합니다. 사용자는 실행 불능 뷰가 대체될 때 경고(SQLSTATE 01595)를 받게 됩니다. 응용프로그램이 바인드 옵션 SQLWARN을 NO로 설정하여 바인드한 경우 경고가 리턴되지 않습니다.

column-name

뷰의 컬럼을 명명합니다. 컬럼 이름의 목록이 지정되면, 이 목록에 포함된 이름 수는 fullselect의 결과 테이블에 있는 컬럼 수와 동일해야 합니다. 각 *column-name*은 고유해야 하고 규정화되어서는 안 됩니다. 컬럼 이름의 목록이 지정되지 않으면, 뷰의 컬럼은 fullselect의 결과 컬럼에 대한 이름을 계승합니다.

fullselect의 결과 테이블에 중복된 컬럼 이름이나 알려져 있지 않은 컬럼이 있는 경우, 컬럼 이름 목록을 반드시 지정해야 합니다(SQLSTATE 42908). 명명되지 않은 컬럼은 컨테이너, 함수, 표현식 또는 세트 조작으로부터 유래된 컬럼으로서, 선택 목록의 AS절을 사용하여 명명되지 않았습니다.

OF *type-name*

뷰의 컬럼들이 *type-name*에 의해 식별되는 구조화 유형 속성에 기초하도록 지정합니다. 스키마 이름 없이 *type-name*이 지정되면, 유형 이름은 SQL 경로(정적 SQL인 경우에는 FUNCPATH 사전 처리 옵션, 동적 SQL인 경우에는 CURRENT PATH 레지스터에 의해 정의)에서 스키마를 검색하여 분석됩니다. 유형 이름은 기존 사용자 정의 유형의 이름이어야 하며(SQLSTATE 42704) 인스턴스 작성 가능한 구조화 유형이어야 합니다(SQLSTATE 428DP).

MODE DB2SQL

이 절은 입력된 뷰의 모드를 지정하는 데 사용됩니다. 이것은 현재 지원되는 유일한 유효한 모드입니다.

UNDER *superview-name*

이 뷰가 *superview-name*의 서브뷰임을 나타냅니다. 상위 뷰는 기존의 뷰여야 하고(SQLSTATE 42704), 이 뷰는 *type-name*의 바로 위 상위 유형인 구조화 유형을 사용하여 정의되어야 합니다(SQLSTATE 428DB). *view-name* 및 *superview-name*의 스키마 이름은 같아야 합니다(SQLSTATE 428DQ).

*superview-name*에 의해 식별되는 뷰에는 *type-name*을 사용하여 이미 정의된 기존의 부속 뷰가 없어야 합니다(SQLSTATE 42742).

뷰의 컬럼에는 REF(*type-name*)로 유형이 수정될 상위 뷰의 오브젝트 식별자 컬럼이 포함됩니다. 이 뒤에는 *type-name* 속성에 기초한 컬럼이 옵니다.(유형에는 상위 유형의 속성이 포함되어야 합니다.)

INHERIT SELECT PRIVILEGES

상위 뷰에 대한 SELECT 특권을 보유하고 있는 사용자 또는 사용자 그룹에게는 새로 작성된 부속 뷰에도 그에 상응하는 특권이 권한 부여됩니다. 부속 뷰 정의자가 이 특권의 권한을 준 사용자로 간주됩니다.

OID-column

입력된 뷰의 오브젝트 식별자 컬럼을 정의합니다.

REF IS *OID-column-name* **USER GENERATED**

오브젝트 식별자(OID) 컬럼이 첫번째 컬럼으로서 뷰에 정의되도록 지정합니다. 뷰 계층의 루트 뷰에 OID가 필수적입니다(SQLSTATE 428DX). 뷰는 부속 뷰가 아닌 입력된 뷰여야 합니다(OF절이 존재해야 합니다)(SQLSTATE 42613). 컬럼 이름은 *OID-column-name*으로서 정의되고, 구조화 유형 *type-name*의 속성 이름과 같아서는 안 됩니다(SQLSTATE 42711). *fullselect*에서 지정된 첫번째 컬럼은 REF(*type-name*) 유형이어야 합니다.(적절한 키를 가지도록 이를 유형변환(cast)해야 할 수도 있습니다.) UNCHECKED가 지정되지 않으면, 색인(기본 키, 고유 제한조건, 고유 색인 또는 0ID 컬럼)을 통해 고유성이 강요되는 널(NULL) 입력 가능 컬럼이 기초가 되어야 합니다. 이 컬럼을 *오브젝트 식별자 컬럼* 또는 *OID 컬럼*이라고 합니다. 키워드 USER GENERATED는 행을 삽입할 때 사용자가 OID 컬럼에 대한 초기 값을 제공해야 한다는 것을 나타냅니다. 일단 행이 삽입되면 OID 컬럼은 갱신할 수 없습니다(SQLSTATE 42808).

UNCHECKED

시스템이 이 고유성을 증명할 수 없을 지라도 고유성을 가정하도록 유형화된 뷰 정의의 오브젝트 식별자 컬럼을 정의합니다. 이는 사용자가 데이터가 이 고유성 규칙을 준수하나 시스템이 고유성을 증명하게 허용하는 규칙에는 따르지 않음을 알고 있는 유형화된 뷰 계층으로 정의되고 있는 테이블이나 뷰와 사용하기 위한 것입니다. UNCHECKED 옵션은 여러 계

층이나 리거시 테이블 또는 뷰에 걸치는 범위의 뷰 계층을 위한 필수입니다. UNCHECKED를 지정하여, 사용자는 뷰의 각 행이 반드시 고유 OID를 가지게 할 책임을 집니다. 사용자가 이러한 등록 정보를 보장하지 못하고, 뷰에 중복 OID 값이 있으며, 비 고유 OID 값 중 하나를 수반하는 path-expression 또는 DEREF 연산자가 오류를 내보낼 수도 있습니다(SQLSTATE 21000).

with-options

입력된 뷰의 컬럼에 적용되는 추가 옵션을 정의합니다.

column-name **WITH OPTIONS**

추가 옵션이 지정될 컬럼의 이름을 지정합니다. *column-name*은 뷰의 *type-name*에 정의된(물려받은 것이 아닌) 속성 이름과 일치해야 합니다. 컬럼은 참조 유형(SQLSTATE 42842)이어야 합니다. 상위 뷰에 있는 컬럼일 수 없습니다(SQLSTATE 428DJ). 컬럼 이름은 명령문의 한 WITH OPTIONS SCOPE절에만 나타날 수 있습니다(SQLSTATE 42613).

SCOPE

참조 유형 컬럼 영역을 식별합니다. 참조 해제(dereference) 연산자의 왼쪽 피연산자 또는 DEREF 함수의 인수로서 사용될 예정인 컬럼에 대한 영역이 지정되어야 합니다.

참조 유형 컬럼 영역을 지정하는 것은 (영역을 물려받지 않은 경우) 후속 ALTER VIEW 명령으로 지연될 수 있습니다. 이는 상호 참조 뷰 및 테이블에서는 항상 목표 테이블이나 뷰가 정의될 수 있도록 하기 위한 것입니다. 뷰의 참조 유형 컬럼에 대해 영역이 지정되지 않고 기본 테이블 또는 뷰 컬럼 영역이 지정된 경우, 참조 유형 컬럼에 의해 기본 컬럼의 영역이 물려받습니다. 기본 테이블이나 뷰 컬럼에 영역이 없었던 경우 컬럼은 영역이 지정되지 않은 상태로(unscope) 남아 있게 됩니다. 영역 및 참조 유형 컬럼에 대한 941 페이지의 『주』에서 좀더 자세한 정보를 참조하십시오.

typed-table-name

입력된 테이블의 이름. 이미 테이블이 있어야 하고, 그렇지 않은 경우 작성되는 테이블 이름과 동일해야 합니다(SQLSTATE 42704). *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는

typed-table-name 유형입니다(SQLSTATE 428DM). 값이 *typed-table-name*에 있는 기존 행들을 실제로 참조하는지 확인하기 위해 *column-name*에 있는 기존 값들에 대해 점검을 하지 않습니다.

typed-view-name

입력된 뷰의 이름. 이미 뷰가 있어야 하고, 그렇지 않은 경우 작성되는 뷰 이름과 동일해야 합니다(SQLSTATE 42704). *column-name*의 데이터 유형은 REF(*S*)여야 합니다. 여기서 *S*는 *typed-view-name* 유형입니다(SQLSTATE 428DM). 값이 *typed-view-name*에 있는 기존 행들을 실제로 참조하는지 확인하기 위해 *column-name*에 있는 기존 값들에 대해 점검을 하지 않습니다.

READ ONLY

컬럼을 읽기 전용 컬럼으로 식별합니다. 이 옵션은 상위 뷰 정의가 내재적으로 읽기 전용인 컬럼에 대해 표현식을 지정할 수 있도록 컬럼을 강제로 읽기 전용 컬럼으로 만드는 데 사용됩니다.

AS

뷰 정의를 식별합니다.

WITH *common-table-expression*

뒤에 나올 fullselect과 함께 사용할 공통 테이블 표현식을 정의합니다. 입력된 뷰를 정의할 경우에는 공통 테이블 표현식을 지정할 수 없습니다. 483 페이지의 『공통 테이블 표현식』에서 참조하십시오.

fullselect

뷰를 정의합니다. 언제든지, 뷰는 SELECT문이 실행된 경우 그 결과 행으로 구성됩니다. fullselect는 호스트 변수, 매개변수 표시문자 또는 선언된 임시 테이블을 참조해서는 안됩니다. 그러나 매개변수 작성 뷰는 SQL 테이블 함수로 작성할 수 있습니다. 726 페이지의 『CREATE FUNCTION (SQL 스칼라, 테이블 또는 행)』에서 참조하십시오.

유형화 뷰 및 서브뷰의 경우: *fullselect*는 다음 규칙을 따라야 하며 그렇지 않으면 오류가 리턴됩니다(달리 지정되지 않을 경우 SQLSTATE 428EA).

- fullselect는 NODENUMBER 또는 PARTITION 함수, 결정할 수 없는 함수 또는 외부 조치를 갖도록 정의된 함수에 대한 참조를 포함해서는 안 됩니다(SQLSTATE 428EA).

- 뷰의 본문은 단일 부속 선택 또는 두 개 이상의 부속 선택의 UNION ALL로 이루어져야 합니다. 뷰 본문에 직접 참여하고 있는 부속 선택의 각각을 뷰의 *가지*라 부른다고 합시다. 뷰에는 하나 이상의 가지가 있을 수 있습니다.
- 각 가지의 FROM절은 그 가지의 *기초가 되는* 테이블이나 뷰라고 하는 단일 테이블이나 뷰(반드시 유형화될 필요는 없음)로 구성되어야 합니다.
- 각 가지의 기초가 되는 테이블이나 뷰는 별도의 계층에 있어야 합니다.(즉, 뷰에는 동일한 계층에 있는 기초가 되는 테이블이나 뷰를 가지고 있는 여러 개의 가지가 있을 수 있습니다.)
- 유형화된 뷰 정의의 가지 중 어느 것도 GROUP BY 또는 HAVING을 지정하지 않을 것입니다.
- 뷰 본문에 UNION ALL이 포함되면, 계층의 루트 뷰가 이 OID 컬럼에 대한 UNCHECKED 옵션을 지정해야 합니다.

뷰 및 서브뷰의 계층에서: BR1 및 BR2를 계층의 뷰 정의에 나타나는 가지라고 가정을 합시다. T1을 BR1의 기초가 되는 테이블이나 뷰라 하고, T2를 BR2의 기초가 되는 테이블이나 뷰라고 가정한 후 다음과 같이 하십시오.

- T1과 T2가 동일한 계층에 있지 않으면, 뷰 계층의 루트 뷰가 그 OID 컬럼에 대한 UNCHECKED 옵션을 지정해야 합니다.
- T1과 T2가 동일한 계층에 있는 경우, BR1과 BR2는 그들의 행 세트들이 해체될 수 있는 충분한 술어나 ONLY절을 포함해야 합니다.

EXTEND AS를 사용하여 정의된 유형화 서브뷰의 경우: 서브뷰의 내용에 있는 모든 가지에 대해

- 각 가지의 기초가 되는 테이블은 바로 위 수퍼뷰의 몇몇 기초가 되는 테이블의 서브이어야 합니다.
- SELECT 목록의 표현식은 서브뷰의 계승되지 않은 컬럼에 지정할 수 있어야 합니다(SQLSTATE 42854).

EXTEND없이 AS를 사용하여 정의된 유형화 서브뷰의 경우:

- 서브뷰의 본문에 있는 모든 가지의 경우, SELECT-목록의 표현식이 서브뷰의 상속 및 비 상속 컬럼의 선언된 유형으로 지정 가능해야 합니다 (SQLSTATE 42854).

- 서브뷰에서 주어진 계층에 대한 각 가지의 OID-표현식은 루트 뷰에서 동일한 계층에 대한 가지의 OID-표현식에 상응해야 합니다(유형변환의 경우는 제외).
- 수퍼뷰에서 READ ONLY로 정의되지 않은(내재적 또는 명시적으로) 컬럼에 대한 표현식은 그 서브뷰에 있는 동일한 기초가 되는 계층에 대한 모든 가지에서 동등해야 합니다.

WITH CHECK OPTION

뷰를 통해 삽입되거나 갱신되는 모든 행은 뷰의 정의를 따라야 한다는 제한규정을 지정합니다. 뷰의 정의에 따르지 않는 행은 뷰의 검색 조건을 만족시키지 않는 행입니다.

WITH CHECK OPTION은 뷰가 읽기 전용일 경우 지정하면 안 됩니다(SQLSTATE 42813). WITH CHECK OPTION이 삽입사항을 허용하지 않는 갱신 가능한 뷰에 대해 지정된 경우, 제한조건은 갱신사항에만 적용됩니다.

뷰가 NODENUMBER 또는 PARTITION 함수, 결정할 수 없는 함수 또는 외부 조치를 가진 함수일 경우에는 WITH CHECK OPTION을 지정해서는 안 됩니다(SQLSTATE 42997).

뷰가 입력된 뷰일 경우에는 WITH CHECK OPTION을 지정해서는 안 됩니다(SQLSTATE 42997).

별명이 뷰의 목표를 갱신하는 경우에는 WITH CHECK OPTION이 지정되면 안 됩니다.

WITH CHECK OPTION이 생략되면, 뷰의 정의가 뷰를 사용하는 삽입 또는 갱신 조작을 점검할 때 사용되지 않습니다. 뷰가 직접적으로 또는 간접적으로 WITH CHECK OPTION을 포함하는 다른 뷰에 대해 종속적일 때, 삽입 또는 갱신 조작시 일부 점검은 계속 발생할 수도 있습니다. 뷰의 정의가 사용되지 않으므로, 행은 뷰의 정의를 따르지 않는 뷰를 통해 삽입되거나 갱신됩니다.

CASCADED

뷰 *V*에 있는 WITH CASCADED CHECK OPTION 제한조건은 *V*가 종속되어 있는 갱신 가능 뷰로부터의 제한조건으로 검색 조건을 승계함을 의미합니다. 뿐만 아니라, *V*에 종속되어 있는 모든 갱신 가능 뷰는 이러

한 제한조건의 주제가 됩니다. 그러므로, *V*의 검색 조건과 *V*가 종속되어 있는 각 뷰는 함께 추가되어 *V* 또는 *V*에 종속되어 있는 모든 뷰의 삽입 또는 갱신에 적용되는 제한조건을 이룹니다.

LOCAL

뷰 *V*에 있는 WITH CASCADED CHECK OPTION 제한조건은 *V* 또는 *V*에 종속되어 있는 모든 뷰의 삽입 또는 갱신에 대한 제한조건으로 *V*의 검색 조건이 적용됨을 의미합니다.

CASCADED와 LOCAL 사이의 차이는 다음 예에 표시됩니다. 다음의 갱신 가능 뷰를 고려하십시오(뒤에 오는 테이블의 컬럼 머리말에서 Y 대체).

```
V1 defined on table T
V2 defined on V1 WITH Y CHECK OPTION
V3 defined on V2
V4 defined on V3 WITH Y CHECK OPTION
V5 defined on V4
```

다음 테이블은 삽입 또는 갱신된 행이 검사되는 검색 조건을 보여줍니다.

	Y is LOCAL	**Y is CASCADED**
V1 checked against:	no view	no view
V2 checked against:	V2	V2, V1
V3 checked against:	V2	V2, V1
V4 checked against:	V2, V4	V4, V3, V2, V1
V5 checked against:	V2, V4	V4, V3, V2, V1

기본 CASCADED 옵션을 사용한 WITH CHECK OPTION의 효과를 보여주는 다음과 같은 갱신 가능 뷰를 고려하십시오.

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10
CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION
CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

*V1*을 사용하는 다음의 INSERT문은 *V1*이 WITH CHECK OPTION을 갖고 있지 않고 *V1*이 WITH CHECK OPTION을 갖는 다른 뷰의 영향을 받지 않으므로 제대로 실행됩니다.

```
INSERT INTO V1 VALUES(5)
```

*V2*를 사용하는 다음의 INSERT문은 *V2*가 WITH CHECK OPTION을 갖고 있고 삽입이 *V2*의 정의를 따르지 않는 행을 생성하므로 오류를 발생합니다.

```
INSERT INTO V2 VALUES(5)
```

*V3*을 사용하는 다음의 INSERT문은 *V3*이 WITH CHECK OPTION이 있는 *V2*에 따라 달라지므로 WITH CHECK OPTION이 없더라도 오류가 발생합니다(SQLSTATE 44000).

```
INSERT INTO V3 VALUES(5)
```

*V3*을 사용하는 다음의 INSERT문은 *V3*(*V3*은 WITH CHECK OPTION을 갖고 있지 않음)의 정의에 따르지 않을 경우에도 제대로 실행됩니다. 이것은 WITH CHECK OPTION을 갖고 있지 않는 *V2*의 정의에 따릅니다.

```
INSERT INTO V3 VALUES(200)
```

주

- 아직 존재하지 않는 스키마 이름을 사용하여 뷰를 작성하면 명령문의 권한 부여 ID가 IMPLICIT_SCHEMA 권한을 가질 경우 그 스키마가 내재적으로 작성됩니다. 스키마 소유자는 SYSIBM입니다. 스키마에 대한 CREATEIN 특권은 PUBLIC에 권한 부여됩니다.
- 뷰 컬럼들은 컬럼이 표현식으로부터 파생될 경우를 제외하고 기본 테이블이나 뷰로부터 NOT NULL WITH DEFAULT 속성을 이어받습니다. 행이 갱신 가능한 뷰에 삽입되거나 갱신되면, 이 행은 기본 테이블에 정의되어 있는 경우에 제한조건(기본 키, 참조 무결성 및 점검)에 대해 점검됩니다.
- 정의에서 실행 불능 뷰를 사용할 경우 새로운 뷰는 작성되지 않습니다(SQLSTATE 51024).
- 이 명령문은 선언된 임시 테이블을 지원하지 않습니다(SQLSTATE 42995).
- *삭제 가능 뷰:* 뷰는 다음 조건 모두가 만족되는 경우 *삭제 가능*합니다.
  - 다른 fullselect의 각 FROM 절이 하나의 기본 테이블(OUTER 절 없음), 삭제 가능 뷰(OUTER 절 없음), 삭제 가능 중첩 테이블 표현식 또는 삭제 가능 공통 테이블 표현식(별명을 식별할 수 없음)만을 식별하는 경우
  - 외부 fullselect에 VALUES절이 포함되지 않는 경우

  - 외부 fullselect에 GROUP BY절 또는 HAVING절이 포함되지 않는 경우
  - 외부 fullselect에 선택 목록 내의 컬럼 함수가 포함되지 않는 경우
  - 외부 fullselect에 UNION ALL 예외가 있는 SET 조작(UNION, EXCEPT 또는 INTERSECT)이 포함되지 않습니다
  - UNION ALL의 피연산자에 있는 기본 테이블들이 동일한 테이블이 아니고 각 피연산자가 삭제 가능해야 합니다
  - 외부 fullselect의 선택 목록에 DISTINCT가 포함되지 않는 경우
- *갱신 가능 뷰:* 뷰의 컬럼은 다음 조건 모두가 만족되는 경우 *갱신 가능*합니다.
  - 뷰가 삭제 가능합니다.
  - 컬럼이 기본 테이블의 컬럼을 분석하고(참조 해제 조작을 사용하지 않고) READ ONLY 옵션이 지정되지 않았습니다.
  - 뷰의 fullselect에 UNION ALL이 들어 있는 경우, UNION ALL 피연산자의 모든 해당 컬럼에 정확하게 일치하는 데이터 유형 (길이 또는 정밀도와 스케일을 포함하여)과 일치하는 기본값이 들어 있습니다.

  뷰의 모든 컬럼이 갱신 가능하면, 그 뷰는 *갱신 가능*합니다.
- *삽입 가능 뷰*

  뷰의 모든 컬럼이 갱신 가능하고 뷰의 fullselect가 UNION ALL을 포함하지 않으면, 뷰는 *삽입 가능*합니다.
- *읽기 전용 뷰:* 뷰는 삭제할 수 없는 경우 *읽기 전용*입니다.

  SYSCAT.VIEWS 카탈로그 뷰의 READONLY 컬럼은 뷰가 읽기 전용인 지를 지정합니다.
- 공통 테이블 표현식과 중첩 테이블 표현식은 동일한 규칙을 사용하여 삭제 가능,갱신 가능, 삽입 가능 또는 읽기 전용 여부를 판별합니다.
- *작동 불능 뷰:* 작동 불능 뷰는 SQL문에 더 이상 사용할 수 없는 뷰입니다. 다음의 경우, 뷰는 실행 불능 상태가 됩니다.
  - 뷰 정의가 종속되는 특권이 권한 취소됩니다.
  - 뷰 정의가 종속된 테이블, 별명(nickname), 별명(alias) 또는 함수와 같은 오브젝트는 삭제되지 않습니다.
  - 뷰 정의가 종속되는 뷰는 실행 불능 상태가 됩니다.

– 뷰 정의의 상위 뷰인 뷰(부속 뷰)는 실행 불능 상태가 됩니다.

실제 용어에서, 실행 불능 뷰는 뷰 정의가 우연히 삭제된 뷰입니다. 예를 들어, 별명이 삭제되면, 그 별명을 사용하여 정의된 뷰가 실행 불능 상태가 됩니다. 모든 종속 뷰도 실행 불능 상태가 되고, 뷰에 종속되는 패키지는 더 이상 유효하지 않게 됩니다.

실행 불능 뷰가 명시적으로 재작성되거나 삭제될 때까지, 실행 불능 뷰를 사용하는 명령문은 컴파일될 수 없습니다(SQLSTATE 51024). 이 경우, CREATE ALIAS, CREATE VIEW, DROP VIEW 및 COMMENT ON TABLE문은 예외입니다. 작동 불능 뷰가 명시적으로 삭제될 때까지, 다른 테이블이나 별명을 작성하는 데 규정화된 이름을 사용할 수 없습니다(SQLSTATE 42710).

실행 불능 뷰는 그의 정의 원문을 사용하는 CREATE VIEWS문을 발행하여 재작성됩니다. 이 뷰(view) 정의 원문은 SYSCAT.VIEWS 카탈로그의 TEXT 컬럼에 저장됩니다. 실행 불능 뷰를 재작성할 때, 다른 사용자에게 필요한 특권을 명시적으로 권한 부여하는 것이 필요한 데, 이는 뷰가 실행 불능으로 표시되면 뷰(view)에 있는 모든 권한 부여 레코드가 삭제되기 때문입니다. 실행 불능 뷰를 재작성하기 위해 이를 명시적으로 삭제할 필요가 없음을 주의하십시오. 실행 불능 뷰와 동일한 *뷰 이름*을 사용하여 CREATE VIEW문을 실행하면 실행 불능 뷰가 대체되고 CREATE VIEW문은 경고를 리턴합니다(SQLSTATE 01595).

실행 불능 뷰는 SYSCAT.VIEWS 카탈로그 뷰의 VALID 컬럼에 있는 X와 SYSCAT.TABLES 카탈로그 뷰의 STATUS 컬럼에 있는 X로 나타납니다.

- *특권*

  뷰의 정의자는 뷰를 삭제할 권한뿐 아니라 뷰에 대한 SELECT 특권도 부여받습니다. 뷰의 정의자는 fullselect에서 식별된 모든 기본 테이블, 뷰 또는 별명에 대한 CONTROL 특권이 있거나 SYSADM 또는 DBADM 권한이 있는 경우에만 해당 뷰에 대한 CONTROL 특권을 확보하게 됩니다.

  뷰가 읽기 전용이 아니고 뷰의 정의자에게 기본 오브젝트에 대해 해당하는 특권이 있는 경우, 뷰의 정의자에게는 뷰에 대한 INSERT, UPDATE, 컬럼 레벨 UPDATE 또는 DELETE 특권이 권한 부여됩니다.

  뷰의 정의자는 뷰 작성시 파생되는 특권들이 존재할 경우 특권들을 획득할 수 있습니다. 뷰 정의자는 PUBLIC이 특권을 갖기 때문에 또는 직접 특권을 가져

야 합니다. 연합 서버 별명에서 뷰를 정의할 때에는특권이 고려되지 않습니다. 그러나, 별명에 대한 뷰를 사용할 때에는 사용자의 권한 부여 ID가 별명이 데이터 소스에서 참조하는 테이블이나 뷰에 대해 유효한 선택 특권을 가져야 합니다. 그렇지 않으면, 오류가 리턴됩니다. 뷰 정의자가 구성원인 그룹에서 보유하는 특권은 고려되지 않습니다.

부속 뷰가 작성될 때, 바로 위 상위 뷰에 있는 SELECT 특권은 자동으로 부속 뷰에 권한 부여됩니다.

- ***Scope 및 REF 컬럼***

  뷰 정의의 fullselect에서 참조 유형 컬럼을 선택할 때, 목표 유형과 필요한 범위를 고려하십시오.

  – 필요한 목표 유형과 범위가 기본 테이블이나 뷰와 동일한 경우, 컬럼을 선택하기만 해도 됩니다.

  – 영역을 변경해야 할 경우, WITH OPTIONS SCOPE절을 사용하여 필요한 영역 테이블이나 뷰를 정의하십시오.

  – 참조의 목표 유형을 변경해야 하는 경우, 컬럼을 우선 참조의 표시 유형으로 유형 변환(cast)한 후 새로운 참조 유형으로 변환해야 합니다. 이런 경우의 영역은 WITH OPTIONS SCOPE절을 사용하거나, 유형 변환시 참조 유형으로 지정할 수 있습니다. 예를 들어, REF(TYP1) SCOPE TAB1으로 지정된 컬럼 Y를 선택한다고 가정하십시오. 이를 REF(VTYP1) SCOPE VIEW1으로 정의하고자 합니다. 선택 목록 항목은 다음과 같습니다.

    ```
    CAST(CAST(Y AS VARCHAR(16) FOR BIT DATA) AS REF(VTYP1) SCOPE VIEW1)
    ```

- ***식별 컬럼*** 뷰의 컬럼은 뷰 정의의 fullselect에서 해당 컬럼의 요소가 테이블의 식별 컬럼 이름이거나 기본 테이블의 식별 컬럼 이름에 직접 또는 간접적으로 맵핑되는 뷰의 컬럼 이름인 경우에 식별 컬럼으로 간주됩니다.

  다른 모든 경우에는 뷰의 컬럼이 식별 등록 정보를 가져오지 않습니다. 예를 들면, 다음과 같습니다.

  – 뷰의 선택 목록에는 식별 컬럼 이름의 여러 인스턴스가 포함되는 경우(즉, 두 번 이상 같은 컬럼 선택)

  – 뷰 정의에 하나의 조인이 포함되는 경우

  – 뷰 정의의 컬럼에 식별 컬럼을 참조하는 하나의 표현식이 포함되는 경우

– 뷰 정의에 UNION이 포함되는 경우

뷰 정의의 선택 목록에 직접 또는 간접적으로 기본 테이블의 식별 컬럼 이름이 포함되는 뷰에 삽입시, INSERT문이 직접 기본 테이블의 식별 컬럼을 참조할 때와 같은 규칙이 적용됩니다.

- ***연합 뷰*** 연합 뷰는 fullselect에 별명에 대한 참조를 포함하는 뷰입니다. 그러한 별명의 존재는 작성시 및 조회에서 뷰가 참조되는 경우 뷰에 사용되는 권한 부여 모델을 변경합니다. 별명을 참조하는 뷰가 작성되고 FEDERATED 키워드가 포함되지 않을 경우, 이 뷰에 대한 권한 부여 요구사항이 별명에 대한 참조때문에 달라짐을 나타내는 경고가 발행됩니다.

  별명에는 연관된 DML 특권이 없으며 따라서 뷰가 작성될 때, 뷰 정의자가 별명 또는 기초가 되는 데이터 소스 테이블이나 뷰에 대한 액세스를 가지는의 여부를 판별하기 위한 어떠한 특권 점검도 행해지지 않습니다. 뷰 정의자가 그러한 오브젝트에 대해 최소한 SELECT 특권을 가질 것을 요구하면서 연합 데이터베이스에 있는 테이블이나 뷰에 대한 참조 점검이 평상시대로 처리됩니다.

  연합 뷰가 다음에 조회에서 참조될 때에는, 조회를 발행한 데이터 소스 및 권한 부여 ID(또는 이것이 맵핑되는 원격 권한 부여 ID)에 대한 조회에서 별명 결과는 데이터 소스 테이블이나 뷰를 액세스하기 위해 필요한 특권을 가지고 있어야 합니다. 연합 뷰를 참조하는 조회를 발행하는 권한 부여 ID는 연합 서버에 존재하는 테이블이나 뷰에 대해 어떠한 추가 특권도 필요로 하지 않습니다.

예

예 1: 문자 'MA'로 시작되는 프로젝트 번호(PROJNO)가 있는 행만을 포함하는 PROJECT 테이블에 뷰 MA_PROJ를 작성하십시오.

```
CREATE VIEW MA_PROJ  AS SELECT *
   FROM PROJECT
     WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

예 2: 예 1처럼 뷰를 작성하지만, 프로젝트 번호(PROJNO), 프로젝트 이름(PROJNAME) 및 프로젝트 담당 사원(RESPEMP)에 대한 컬럼만 선택합니다.

```
CREATE VIEW MA_PROJ
AS SELECTPROJNO, PROJNAME, RESPEMP
 FROM PROJECT
     WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

예 3: 예 2처럼 뷰를 작성하지만, 뷰에서 프로젝트 IN_CHARGE를 담당하는 직원에 대한 컬럼을 호출하십시오.

```
CREATE VIEW MA_PROJ
(PROJNO, PROJNAME, IN_CHARGE)
AS SELECTPROJNO, PROJNAME, RESPEMP
 FROM PROJECT
     WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

주: 컬럼 이름 중 하나만 변경되는 경우에도, 뷰에 있는 세 컬럼 모두의 이름이 괄호안에 나열되어야 하며 그 다음에는 MA_PROJ가 와야 합니다.

예 4: 프로젝트를 담당하는 직원(RESPEMP)의 성(LASTNAME)과 함께 PROJECT 테이블의 처음 네 컬럼(PROJNO, PROJNAME, DEPTNO, RESPEMP)을 포함하는 뷰 PRJ_LEADER를 작성하십시오. EMPLOYEE의 EMPNO를 PROJECT의 RESPEMP와 대조하여 EMPLOYEE 테이블로부터 이름을 확보합니다.

```
CREATE VIEW PRJ_LEADER
 AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME
        FROM PROJECT, EMPLOYEE
        WHERE RESPEMP = EMPNO
```

예 5: 예 4처럼 뷰를 작성하지만, PROJNO, PROJNAME, DEPTNO, RESPEMP 및 LASTNAME 컬럼 외에 책임자의 총 급여(SALARY + BONUS + COMM)를 보여줍니다. 또한, 직원이(PRSTAFF)이 둘 이상인 프로젝트만 선택합니다.

```
CREATE VIEW PRJ_LEADER
(PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, TOTAL_PAY )
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, SALARY+BONUS+COMM
        FROM PROJECT, EMPLOYEE
        WHERE RESPEMP = EMPNO
        AND PRSTAFF > 1
```

fullselect에서 표현식 SALARY+BONUS+COMM을 TOTAL_PAYS로 명명하면 컬럼 이름 목록을 지정하지 않아도 됩니다.

```
CREATE VIEW PRJ_LEADER
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP,
                        LASTNAME, SALARY+BONUS+COMM AS TOTAL_PAY
        FROM PROJECT, EMPLOYEE
        WHERE RESPEMP = EMPNO AND PRSTAFF > 1
```

예 6: 다음 그림에 제공된 테이블과 뷰가 표시됩니다.

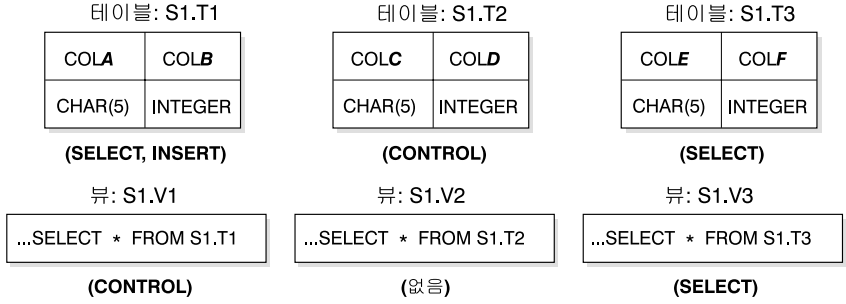


그림 12. 예 6에 대한 테이블과 뷰

사용자 ZORPIE(DBADM 또는 SYSADM 권한을 갖지 않음)에게는 각 오브젝트 아래에 표시된 특권이 권한 부여되었습니다.

1. ZORPIE는 다음에 대해 작성하는 뷰에 대해 CONTROL 특권을 확보하게 됩니다.

   ```
   CREATE VIEW VA AS SELECT * FROM S1.V1
   ```

   이는 사용자에게 S1.V1의 CONTROL이 있기 때문입니다. [86] 기초가 되는 기본 테이블에 대해 갖는 특권이 무엇인 지는 문제가 되지 않습니다.

2. ZORPIE는 뷰를 작성할 때 허용되지 않습니다.

   ```
   CREATE VIEW VB AS SELECT * FROM S1.V2
   ```

   S1.V2에 대해 CONTROL과 SELECT를 갖고 있지 않기 때문입니다. 기본 테이블(S1.T2)에 CONTROL을 갖고 있는 것은 문제가 되지 않습니다.

3. ZORPIE는 다음에 대해 작성하는 뷰에 대해 CONTROL 특권을 확보하게 됩니다.

   ```
   CREATE VIEW VC (COLA, COLB, COLC, COLD)
     AS SELECT * FROM S1.V1, S1.T2
     WHERE COLA = COLC
   ```

   ZORPIE.VC의 fullselect는 뷰 S1.V1과 테이블 S1.T2를 참조하고 이 두 곳에 CONTROL을 갖고 있기 때문입니다. 뷰 VC는 읽기 전용이므로, ZORPIE는 INSERT, UPDATE 또는 DELETE 특권을 갖지 않습니다.

4. ZORPIE는 다음에 대해 작성하는 뷰에 대해 SELECT 특권을 확보하게 됩니다.

86. S1.V1의 CONTROL은 DBADM 또는 SYSADM 권한이 있는 사용자에 의해 ZORPIE에 권한 부여되었어야 합니다.

```
CREATE VIEW VD (COLA,COLB, COLE, COLF)
  AS SELECT * FROM S1.V1, S1.V3
  WHERE COLA = COLE
```

ZORPIE.VD의 fullselect가 SELECT 특권만 있는 뷰와 CONTROL 특권이 있는 뷰인 두가지 뷰 S1.V1 및 S1.V3을 참조하기 때문입니다. ZORPIE.VD에서의 두 특권, SELECT 중 덜 중요한 것이 부여됩니다.

5. ZORPIE는 GRANT 옵션과 더불어 INSERT, UPDATE 및 DELETE 특권과 아래의 뷰 정의에 있는 VE 뷰에 대해 SELECT 특권을 갖게 됩니다.

```
CREATE VIEW VE AS SELECT * FROM S1.V1
WHERE COLA > ANY
       (SELECT COLE FROM S1.V3)
```

VE에서 ZORPIE의 특권은 S1.V1에서의 특권에 따라 1차적으로 결정됩니다. S1.V3는 부속 조회에서만 참조되므로, VE 뷰를 작성하려면 S1.V3에 대해 SELECT 특권만 있으면 됩니다. 뷰의 정의자는 뷰 정의시 참조되는 모든 오브젝트에 대해 CONTROL을 갖는 경우 뷰에 대해 CONTROL을 얻게 됩니다. ZORPIE는 S1.V3에 대해 CONTROL을 갖지 않으므로 VE에 대해 CONTROL을 갖지 않습니다.

CREATE WRAPPER

CREATE WRAPPER문은 랩퍼(연합 서버가 데이터 소스의 특정 범주와 상호작용하는 데 사용되는 메카니즘)를 연합 서버에 등록합니다.

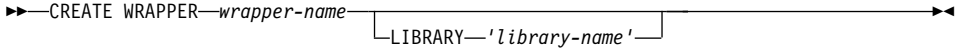
호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID는 SYSADM 또는 DBADM 권한이 있어야 합니다.

구문



설명

wrapper-name

랩퍼를 명명합니다. 다음과 같은 이름이 가능합니다.

- 사전 정의된 이름. 사전 정의된 이름이 지정되면, 연합 서버가 자동으로 기본값을 *'library-name'*으로 지정합니다.

  사전 정의된 이름은 다음과 같습니다.

  DRDA — 모든 DB2 계열 데이터 소스의 경우

  NET8 — Oracle의 Net8 클라이언트 소프트웨어에 의해 지원되는 모든 Oracle 데이터 소스의 경우

  OLEDB — Microsoft OLE DB에 의해 지원되는 모든 OLE DB 제공자의 경우

SQLNET Oracle의 SQL*Net 클라이언트 소프트웨어에 의해 지원되는 모든 Oracle 데이터 소스의 경우

- 사용자 제공 이름. 그러한 이름이 제공되면, *'library-name'* 역시 지정해야 합니다.

LIBRARY *'library-name'*

랩퍼 모듈을 포함하는 파일에 이름을 부여합니다. LIBRARY 옵션은 사용자 제공 *wrapper-name*이 사용된 경우에만 필요합니다. 이 옵션은 사전정의된 *wrapper-name*이 제공된 경우에는 사용할 수 없습니다. 사전 정의된 *wrapper-names*에 대한 기본 파일 이름은 다음과 같습니다.

표 26. LIBRARY 옵션에 대한 기본 파일 이름

플랫폼	DRDA	SQLNET	NET8	OLEDB
AIX	**libdrda.a**	**libsqlnet.a**	**libnet8.a**	-
HP-UX	**libdrda.sl**	**libsqlnet.sl**	**libnet8.sl**	-
SOLARIS	**libdrda.so**	**libsqlnet.so**	**libnet8.so**	-
UNIX	**libdrda.a**	**libsqlnet.a**	**libnet8.a**	-
WINNT	**drda.dll**	**sqlnet.dll**	**net8.dll**	**db2oledb.dll**

주

랩퍼를 선택하고 정의하는 방법에 대해서는 *설치 및 구성 보충 설명서*에서 자세한 정보를 참조하십시오.

예

예 1: 연합 서버가 Oracle의 SQL*Net 클라이언트 소프트웨어에 의해 지원되는 Oracle 데이터 소스와 상호작용할 수 있는 랩퍼 등록. 사전 정의된 이름을 사용하십시오.

```
CREATE WRAPPER SQLNET
```

예 2: AIX 시스템상의 연합 서버가 VM 및 VSE용 DB2 데이터 소스와 상호작용하는 데 사용할 수 있는 랩퍼를 등록하십시오. 테스트에 사용된 데이터 소스를 가리키는 이름을 지정합니다.

```
CREATE WRAPPER TEST
   LIBRARY 'libsqlds.a'
```

라이브러리 이름에 있는 확장자(a)는 랩퍼 TEST가 AIX 시스템에 상주하는 데이터 소스에 대한 것임을 나타냅니다.

DECLARE CURSOR

DECLARE CURSOR문은 커서를 정의합니다.

호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 포함될 수 있습니다. 이 명령문은 실행 가능 명령문이 아니며 동적으로 준비될 수 없습니다.

권한 부여

이 용어는 "커서의 SELECT문"은 권한 부여 규칙을 지정하는 데 사용됩니다. 커서의 SELECT문은 다음 중 하나입니다.

- *명령문 이름*에 의해 식별된 준비된 select
- 지정된 *select*문

커서의 SELECT문에서 식별된(직접적으로 또는 별명을 사용하여) 각 테이블 또는 뷰(view)마다, 명령문의 권한 부여 ID로 보유되는 특권에는 최소한 다음 중 하나가 포함되어야 합니다.

- SYSADM 또는 DBADM 권한
- *select*문에 식별되는 각 테이블이나 뷰에 대해
  - 테이블이나 뷰에서의 SELECT 특권 또는
  - 테이블이나 뷰에서의 CONTROL 특권

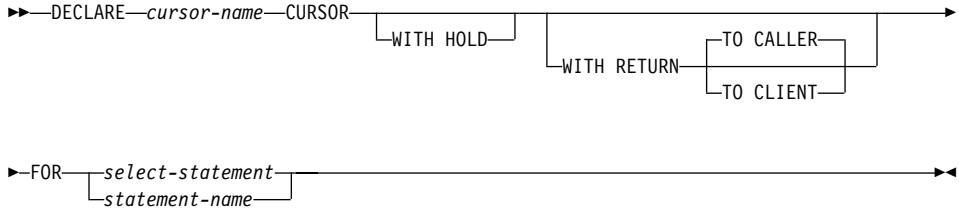
***statement-name*이 지정된 경우,**

- 명령문의 권한 부여 ID는 런타임 권한 부여 ID입니다.
- 권한 점검은 select문이 준비될 때 수행됩니다.
- 커서는 select문이 올바로 준비되지 않으면 열리지 않습니다.

***select*문이 지정된 경우,**

- GROUP 특권이 점검되지 않습니다.
- 명령문의 권한 부여 ID는 프로그램 준비시 지정되는 권한 부여 ID입니다.

구문



설명

cursor-name

원시 프로그램이 수행될 때 작성된 커서의 이름을 지정합니다. 이름은 원시 프로그램에서 선언된 다른 커서의 이름과 동일해선 안 됩니다. 커서는 사용하기 전에 열려야 합니다(1081 페이지의 『OPEN』 참조).

WITH HOLD

여러 작업 단위에서 자원을 유지보수합니다. WITH HOLD 커서 속성의 영향은 다음과 같습니다.

- COMMIT로 끝나는 작업 단위(UOW)의 경우:
  - WITH HOLD로 정의된 열린 커서는 열린 채로 있습니다. 커서는 결과 테이블의 다음 논리 행 전에 위치됩니다.

    DISCONNECT문이 WITH HOLD 커서를 갖는 연결에 대해 COMMIT문 다음에 발행된 경우, 보유된 커서는 명시적으로 닫혀 있어야 합니다. 그렇지 않으면, 연결은 작업을 수행한 것(SQL문이 발행되지 않았어도 열린 WITH HELD 커서를 지님으로써)으로 간주되어 DISCONNECT문이 실패하게 됩니다.
  - 열려 있는 WITH HOLD 커서의 현재 커서 위치를 보호하기 위한 잠금을 제외한 모든 잠금이 해제됩니다. 테이블에 대한 잠금과, 병렬 환경의 경우 커서가 현재 위치한 행에 대한 잠금은 그대로 유지됩니다. 패키지와 동적 SQL 섹션(있을 경우)에 대한 잠금은 그대로 유지됩니다.
  - 다음은 COMMIT 요청 바로 다음에 오는 WITH HOLD 정의되는 커서에 대한 유효한 조작입니다.
    - FETCH: 커서의 다음 행을 페치합니다.

- CLOSE: 커서를 닫습니다.

– UPDATE 및 DELETE CURRENT OF CURSOR는 같은 작업 단위 내에서 패치되는 행에 대해서만 유효합니다.

– LOB 위치 지정자는 해제됩니다.

• ROLLBACK로 끝나는 작업 단위의 경우:

– 열린 커서는 모두 닫힙니다.

– 작업 단위에서 획득된 모든 잠금은 해제됩니다.

– LOB 위치 지정자는 해제됩니다.

• 특수 COMMIT의 경우:

– 패키지는 패키지를 바인딩하여 명시적 또는 암시적으로 재작성될 수 있습니다. 패키지의 유효성이 검사된 후 처음으로 참조될 때 동적으로 재작성되었기 때문입니다. 패키지의 리바인드중 보유된 모든 커서가 닫힙니다. 이는 후속 실행에서 오류를 야기할 수도 있습니다.

WITH RETURN

이 절은 커서가 저장 프로시듀어로부터의 결과 세트로 사용됨을 나타냅니다. WITH RETURN은 DECLARE CURSOR문이 저장 프로시듀어의 소스 코드와 함께 포함되는 경우에만 관련됩니다. 다른 경우에는 사전 처리 컴파일러가 이 절을 사용할 수 있으나 아무 효과도 없습니다.

SQL 프로시듀어내에서 SQL 프로시듀어 종료시에도 여전히 열려 있는 WITH RETURN절을 사용하여 선언된 커서는 SQL 프로시듀어로부터의 결과 세트를 정의합니다. SQL 프로시듀어내의 열려 있는 다른 모든 커서는 SQL 프로시듀어가 종료될 때 닫힙니다. 외부 저장 프로시듀어(LANGUAGE SQL을 사용하여 정의되지 않은)내에서는 WITH RETURN절이 아무 효과도 없으며 외부 프로시듀어의 끝에 열려 있는 모든 커서는 결과 세트로 간주됩니다.

TO CALLER

커서가 결과 세트를 호출자에게 리턴할 수 있도록 지정합니다. 예를 들어 호출자가 다른 저장 프로시듀어인 경우 결과 세트는 해당 저장 프로시듀어로 리턴됩니다. 호출자가 클라이언트 응용프로그램인 경우 결과 세트는 클라이언트 응용프로그램으로 리턴됩니다.

TO CLIENT

커서가 결과 세트를 클라이언트 응용프로그램으로 리턴할 수 있도록 지정합니다. 이 커서는 중간 중첩된 프로시듀어에게는 보이지 않습니다.

select-statement

커서의 SELECT문을 식별합니다. *select-statement*은 매개변수 표시문자를 포함해서는 안되나, 호스트 변수에 대한 참조는 포함할 수 있습니다. 호스트 변수의 선언은 원시 프로그램에서 DECLARE CURSOR문 앞에 있어야 합니다. *select-statement*의 설명에 대해서는 482 페이지의 『select문』에서 참조하십시오.

statement-name

statement-name 커서의 SELECT문은 커서가 열릴 때 명령문 이름으로 식별되는 준비된 SELECT문입니다. *statement-name*은 원시 프로그램의 다른 DECLARE CURSOR문에 지정된 *statement-name*과 같으면 안 됩니다.

준비된 SELECT문에 대한 1087 페이지의 『PREPARE』에서 설명을 참조하십시오.

주

- 다른 프로그램 또는 같은 프로그램내의 다른 소스 파일로부터 호출되는 프로그램은 호출하는 프로그램이 열어 놓은 커서를 사용할 수 없습니다.
- SQL이 아닌 다른 LANGUAGE로 중첩되지 않은 저장 프로시듀어는 WITH RETURN절없이 DECLARE CURSOR가 지정되고 커서가 프로시듀어에서 열려 있는 경우 기본 동작으로 WITH RETURN TO CALLER를 가지게 됩니다. 이는 이전 버전의 저장 프로시듀어로 하여금 리턴 결과 세트를 적용 가능한 클라이언트 응용프로그램으로 리턴할 수 있는 호환성을 제공합니다. 이러한 동작을 피하려면 해당 프로시듀어에서 열려 있는 모든 커서를 닫으십시오.
- 커서의 SELECT문에 CURRENT DATE, CURRENT TIME 또는 CURRENT TIMESTAMP가 포함되어 있으면, 이 특수 레지스터에 대한 모든 참조사항은 각 FETCH에서 같은 값을 생성합니다. 이 값은 커서가 열릴 때 결정됩니다.
- 데이터의 효율적인 처리를 위해, 데이터베이스 관리자는 원격 서버로부터 데이터를 검색할 때 읽기 전용 커서에 대해 데이터를 블록화할 수 있습니다. FOR UPDATE절을 사용하면, 데이터베이스 관리 프로그램에서 커서가 갱신 가능 여

부를 쉽게 결정할 수 있습니다. 갱신 가능성은 액세스 경로 선택을 결정할 경우에도 사용됩니다. 커서가 위치지정된 UPDATE나 DELETE문에서 사용되지 않을 경우 FOR READ ONLY로 선언되어야 합니다.

- 열린 상태의 커서는 결과 테이블과 그 테이블의 행에 상대적인 위치를 지시합니다. 테이블은 커서의 SELECT문에 의해 지정된 결과 테이블입니다.
- 커서는 다음 조건 모두가 만족되는 경우 *삭제 가능*합니다.
  - 외부 fullselect의 각 FROM 절이 OUTER 절을 사용하지 않고 하나의 기본 테이블이나 삭제 가능 뷰를 식별하는 경우(중첩된 또는 공통 테이블 표현식이나 별명은 식별할 수 없습니다)
  - 외부 fullselect에 VALUES절이 포함되지 않는 경우
  - 외부 fullselect에 GROUP BY절 또는 HAVING절이 포함되지 않는 경우
  - 외부 fullselect에 선택 목록 내의 컬럼 함수가 포함되지 않는 경우
  - 외부 fullselect에 UNION ALL 예외가 있는 SET 조작(UNION, EXCEPT 또는 INTERSECT)이 포함되지 않는 경우
  - 외부 fullselect의 선택 목록에 DISTINCT가 포함되지 않는 경우
  - 선택문에 ORDER BY절이 포함되지 않는 경우
  - SELECT 문에 FOR READ ONLY 절이 포함되지 않는 경우[87]
  - 다음 중 하나 이상이 만족되는 경우:
    - FOR UPDATE절[88]이 지정됩니다.
    - 커서가 정적으로 정의되었습니다.
    - LANGLEVEL 바인드 옵션이 MIA 또는 SQL92E인 경우

  커서와 연관된 외부 fullselect의 선택 목록에 있는 컬럼은 다음 모두가 만족되는 경우 *갱신 가능*합니다.
  - 커서가 삭제 가능한 경우
  - 컬럼이 기본 테이블의 컬럼을 해석하는 경우

87. FOR READ ONLY 절은 491 페이지의 『read-only절』에 정의되어 있습니다.

88. FOR UPDATE절은 490 페이지의 『update절』에서 정의됩니다.

– LANGLEVEL 바인드 옵션이 MIA이고, SQL92E 또는 선택문에 FOR UPDATE 절이 포함되는 경우(해당 컬럼을 FOR UPDATE 절에 명시적으로 또는 내재적으로 지정해야 합니다).

커서가 삭제 불가능한 경우, *읽기 전용*입니다.

커서는 다음 조건 모두가 만족되는 경우 *불명확한* 커서입니다.

– 선택문이 동적으로 준비된 경우

– 선택문에 FOR READ ONLY 절이나 FOR UPDATE 절이 포함되지 않는 경우

– LANGLEVEL 바인드 옵션이 SAA1인 경우

– 그렇지 않으면, 커서가 삭제 가능 커서의 조건을 만족하는 경우

불명확한 커서는 BLOCKING 바인드 옵션이 ALL인 경우 읽기 전용으로 간주되고, 그렇지 않으면, 삭제 가능으로 간주됩니다.

- CLI를 사용하여 작성된 응용프로그램이 호출한 저장 프로시듀어의 커서는 클라이언트 응용프로그램으로 직접 리턴된 결과 집합을 정의하는 데 사용됩니다. SQL 프로시듀어의 커서는 WITH RETURN 절을 사용하여 정의된 경우에만 호출하는 SQL 프로시듀어로 리턴될 수 있습니다. 583 페이지의 『주』에서 참조하십시오.

예

DECLARE CURSOR문은 커서 이름 C1을 SELECT의 결과와 연관시킵니다.

```
EXEC SQL  DECLARE C1 CURSOR FOR
    SELECT DEPTNO, DEPTNAME, MGRNO
  FROM DEPARTMENT
    WHERE ADMRDEPT = 'A00';
```

DECLARE GLOBAL TEMPORARY TABLE

DECLARE GLOBAL TEMPORARY TABLE문은 현재 세션의 임시 테이블을 정의합니다. 선언된 임시 테이블 설명은 시스템 카탈로그에 나타나지 않습니다. 임시 테이블은 영속적이 아니며 다른 세션과 공유할 수 없습니다. 같은 이름의 선언된 전역 임시 테이블을 정의하는 각 세션은 그 소유의 고유한 임시 테이블 설명을 가집니다. 세션이 종료되면 테이블의 행이 삭제되고 임시 테이블의 설명이 삭제됩니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

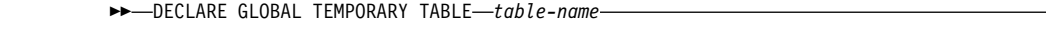
명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

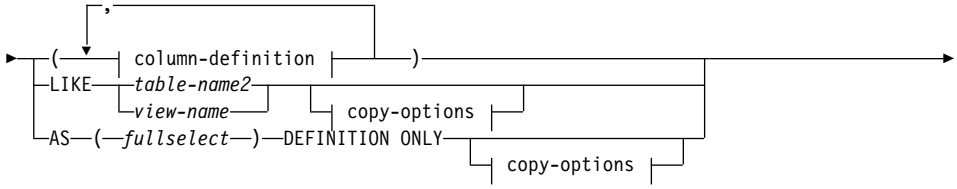
- SYSADM 또는 DBADM 권한
- USER TEMPORARY 테이블 공간에서의 USE 특권

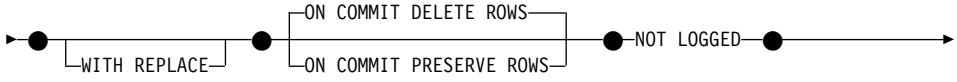
LIKE 또는 fullselect를 사용하여 테이블을 정의하는 경우, 명령문의 권한 부여 ID가 보유하는 특권은 각각의 식별된 테이블이나 뷰에 대해 최소한 다음 중 하나를 포함해야 합니다.

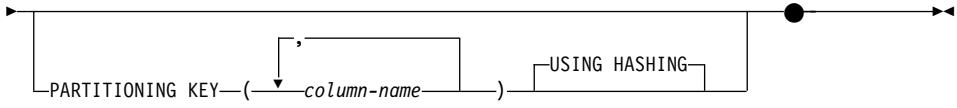
- 테이블 또는 뷰에서의 SELECT 특권
- 테이블 또는 뷰에서의 CONTROL 특권
- SYSADM 또는 DBADM 권한

구문

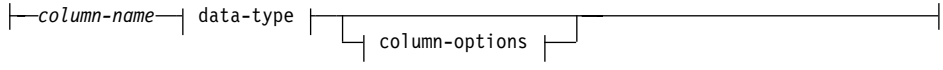




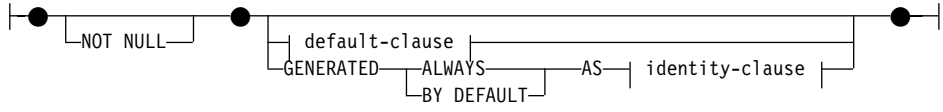




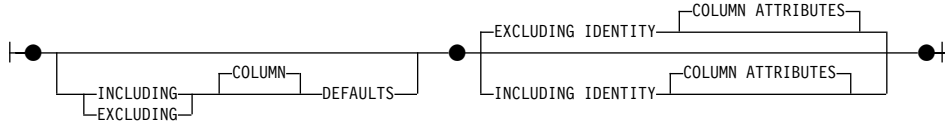
column-definition:



column-options:



copy-options:



설명

table-name

임시 테이블을 명명합니다. 명시적으로 지정된 경우 규정자는 SESSION이어

야 하며, 그렇지 않으면 오류가 리턴됩니다(SQLSTATE 428EK). 규정자가 지정되지 않은 경우에는 SESSION이 내재적으로 지정됩니다.

같은 *table-name*의 선언된 전역 임시 테이블을 정의하는 각 세션은 그 고유의 해당하는 선언된 전역 임시 테이블 설명을 가집니다. *table-name*이 이미 세션에 있는 선언된 임시 테이블을 식별하는 경우 WITH REPLACE 절을 지정해야 합니다(SQLSTATE 42710).

같은 이름을 가지며 스키마 이름은 SESSION인 테이블, 뷰, 별명(alias) 또는 별명(nickname)이 이미 카탈로그에 있을 가능성이 있습니다. 이 경우,

- 선언된 전역 임시 테이블 *table-name*는 오류나 경고없이 여전히 정의되어 있을 수 있습니다.
- SESSION.*table-name*에 대한 참조는 카탈로그에 이미 정의되어 있는 SESSION.*table-name*이 아닌 선언된 전역 임시 테이블로 해석됩니다.

column-definition

임시 테이블의 컬럼 속성을 정의합니다.

column-name

테이블의 컬럼을 명명합니다. 이름을 규정화할 수 없으며, 테이블의 두 개 이상 컬럼에 대해 동일한 이름을 사용할 수 없습니다(SQLSTATE 42711).

테이블에는 다음 사항이 있습니다.

- 최대 500 컬럼이 있는 4K 페이지 크기. 여기서, 컬럼의 바이트 수는 4K 페이지 크기 단위로 4005보다 커서는 안 됩니다. 853 페이지의 『행 크기』에서 자세한 내용을 참조하십시오.
- 최대 1 012 컬럼을 가진 8K 페이지 크기. 여기서, 컬럼의 바이트 수는 8 101 보다 커서는 안 됩니다. 853 페이지의 『행 크기』에서 자세한 내용을 참조하십시오.
- 최대 1 012 컬럼을 가진 16K 페이지 크기. 여기서, 컬럼의 바이트 수는 16 293 보다 커서는 안 됩니다. 853 페이지의 『행 크기』에서 자세한 내용을 참조하십시오.
- 최대 1 012 컬럼을 가진 32K 페이지 크기. 여기서, 컬럼의 바이트 수는 32 677 보다 커서는 안 됩니다. 853 페이지의 『행 크기』에서 자세한 내용을 참조하십시오.

data-type

허용 가능 유형에 대해서는 800 페이지의 『CREATE TABLE』의 *data-type*을 참조하십시오. BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, 참조 및 구조화 유형은 선언된 전역 임시 테이블과 함께 사용할 수 없습니다(SQLSTATE 42962). 이러한 제한된 유형에 근거하는 구별 유형은 예외입니다.

FOR BIT DATA는 문자열 데이터 유형의 일부로 지정할 수 있습니다.

column-options

테이블 컬럼에 관련된 추가 옵션을 정의합니다.

NOT NULL

컬럼이 널(NULL) 값을 포함하지 않도록 합니다. 널(NULL) 값 스펙에 대해서는 800 페이지의 『CREATE TABLE』의 *NOT NULL*을 참조하십시오.

default-clause

기본값 스펙에 대해서는 800 페이지의 『CREATE TABLE』의 *default-clause*를 참조하십시오.

identity-clause

식별 컬럼 스펙에 대해서는 800 페이지의 『CREATE TABLE』의 *identity-clause*를 참조하십시오.

LIKE *table-name2* **또는** *view-name*

테이블 컬럼이 식별된 테이블(*table-name2*) 또는 뷰(*view-name*) 또는 별명의 컬럼과 정확하게 같은 이름 및 설명을 가지도록 지정합니다. LIKE 다음에 지정된 이름은 카탈로그 또는 선언된 임시 테이블에 있는 테이블, 뷰 또는 별명을 식별해야 합니다. 유형화 테이블 또는 유형화 뷰는 지정할 수 없습니다(SQLSTATE 428EC).

LIKE의 사용은 *n* 컬럼의 내재된 정의입니다. 여기서 *n*은 식별된 테이블 또는 뷰의 컬럼 수입니다.

- 식별된 테이블인 경우, 내재된 정의에는 *table-name2*의 각 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(null) 입력 가능 특성이 포함됩니다. EXCLUDING COLUMN DEFAULTS가 지정되지 않을 경우에는 컬럼 기본값도 포함됩니다.
- 식별된 뷰인 경우, 내재된 정의에는 *view-name*에 정의된 fullselect의 각 결과 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(NULL) 입력 가능 특성이 포함됩니다.

복사 속성 절을 기초로 컬럼 기본값 및 식별 컬럼 속성을 포함하거나 제외시킬 수 있습니다.

내재된 정의에는 식별된 테이블 또는 뷰의 다른 어느 속성도 포함되지 않습니다. 그러므로 새로운 테이블이 고유한 제한조건, 외부 키 제한조건, 트리거 또는 색인을 가지지 않습니다. 테이블은 IN 절에 지정된 대로 내재적으로 또는 명시적으로 테이블 공간에 작성됩니다.

table-name2 및 *view-name*에 사용된 이름은 작성 중인 전역 임시 테이블의 이름과 같을 수 없습니다(SQLSTATE 428EC).

AS (*fullselect*) DEFINITION ONLY

테이블 정의가 조회 표현식 결과인 컬럼 정의를 기초로 함을 지정합니다. AS(*fullselect*)의 사용은 *n* 컬럼의 내재된 정의입니다. 여기서 *n*은 *fullselect* 결과 컬럼의 수입니다. 새 테이블의 컬럼은 *fullselect*의 결과인 컬럼으로 정의됩니다. 모든 선택 목록 요소는 고유한 이름이어야 합니다(SQLSTATE 42711). AS 절은 SELECT 절에서 고유한 이름을 제공하는 데 사용할 수 있습니다.

내재된 정의에는 *fullselect*에 각 결과 컬럼에 대한 컬럼 이름, 데이터 유형 및 널(NULL) 입력 가능 특성이 포함됩니다.

copy-options

이러한 옵션은 소스 결과 테이블 정의(테이블, 뷰 또는 fullselect)의 추가 속성을 복사할지 여부를 지정합니다.

INCLUDING COLUMN DEFAULTS

소스 결과 테이블 정의의 갱신 가능 컬럼 각각에 대한 컬럼 기본값이 복사됩니다. 갱신 가능하지 않은 컬럼은 작성된 테이블의 해당 컬럼에 정의된 기본값을 가지지 않게 됩니다.

LIKE *table-name2*가 지정되고 *table-name2*가 기본 테이블 또는 선언된 임시 테이블을 식별하는 경우에는 INCLUDING COLUMN DEFAULTS가 기본값입니다.

EXCLUDING COLUMN DEFAULTS

컬럼 기본값은 소스 결과 테이블 정의로부터 복사되지 않습니다.

LIKE *table-name*이 지정되고 *table-name*이 기본 테이블 또는 선언된 임시 테이블을 식별하는 경우를 제외하고는 이 절이 기본값입니다.

INCLUDING IDENTITY COLUMN ATTRIBUTES

가능한 경우 식별 컬럼 속성(START WITH, INCREMENT BY 및 CACHE 값)이 소스 결과 테이블 정의로부터 복사됩니다. 테이블, 뷰 또는 fullselect의 해당 컬럼 요소가 테이블 컬럼의 이름이거나 직접 또는 간접적으로 식별 등록 정보를 가지는 기본 테이블 컬럼의 이름에 해당되는 뷰 컬럼의 이름인 경우 식별 컬럼 속성을 복사할 수 있습니다. 다른 모든 경우에는 새 임시 테이블의 컬럼이 식별 등록 정보를 가져오지 않습니다. 예를 들면, 다음과 같습니다.

- fullselect의 선택 목록에는 식별 컬럼 이름의 여러 인스턴스가 포함됩니다(즉, 두 번 이상 같은 컬럼 선택).
- fullselect의 선택 목록에 여러 식별 컬럼이 포함됩니다(즉, 하나의 조인 포함).
- 식별 컬럼이 선택 목록의 표현식에 포함됩니다.
- fullselect에 하나의 설정 연산(union, except 또는 intersect)이 포함됩니다.

EXCLUDING IDENTITY COLUMN ATTRIBUTES

식별 컬럼 속성이 소스 결과 테이블 정의로부터 복사되지 않습니다.

ON COMMIT

COMMIT 조작 수행시 전역 임시 테이블에 대해 취한 조치를 지정합니다.

DELETE ROWS

테이블에 대해 열려 있는 WITH HOLD 커서가 없는 경우 테이블의 모든 행이 삭제됩니다. 이것은 기본값입니다.

PRESERVE ROWS

테이블 행은 유지됩니다.

NOT LOGGED

테이블 작성을 비롯한 테이블 변경사항이 기록되지 않았습니다. ROLLBACK(또는 ROLLBACK TO SAVEPOINT) 조작이 수행되고 작업 단위(UOW)(또는 세이브포인트)에서 테이블이 변경되면 테이블의 모든 행이 삭제됩니다. 테이블이 작업 단위(또는 세이브포인트)에 작성되었다면 해당 테이블이 삭제됩니다. 테이블이 작업 단위(또는 세이브포인트)에서 삭제되었다면 해당 테이블이 복원되기는 하지만 행은 없습니다. 추가로 해당 테이블에 대해 INSERT, UPDATE 또는 DELETE 조작을 수행하는 명령문이 오류를 발견하게 되면 테이블의 모든 행이 삭제됩니다.

WITH REPLACE

지정된 이름의 선언된 전역 임시 테이블이 이미 있는 경우 기존 테이블은 이 명령문으로 정의된 임시 테이블로 대체됩니다(그리고 기존 테이블의 모든 행은 삭제됩니다).

WITH REPLACE가 지정되지 않았다면, 지정된 이름이 이미 현재 세션에 있는 선언된 전역 임시 테이블을 식별해서는 안됩니다(SQLSTATE 42710).

IN *tablespace-name*

전역 임시 테이블의 인스턴스가 작성될 테이블 공간을 식별합니다. 테이블 공간이 있어야 하고 이 테이블 공간은 명령문의 권한 부여 ID가 USE 특권을 가지는(SQLSTATE 42501) 사용자 임시 테이블 공간이어야 합니다(SQLSTATE 42501). 이 절이 지정되지 않은 경우, 해당 명령문의 권한 부여 ID가 USE 특권을 가지는 가장 작은 충분한 페이지 *크기*의 사용자 임시 테이블 공간을 선택함으로써 이 테이블의 테이블 공간을 결정합니다. 둘 이상의 테이블 공간이 규정되었다면 USE 특권이 부여된 사용자에 따라 환경설정이 제공됩니다.

1. 권한 부여 ID
2. 권한 부여 ID가 포함된 그룹
3. PUBLIC

둘 이상의 테이블 공간이 규정되었다면 데이터베이스 관리 프로그램에서 최종 선택을 합니다. 사용자 임시 테이블 공간이 규정된 경우에는 오류가 발생합니다(SQLSTATE 42727).

테이블 공간 결정은 다음의 경우에 변경할 수 있습니다.

- 테이블 공간이 삭제 또는 작성되었을 때
- USE 특권이 권한 부여 또는 권한 취소되었을 때

테이블의 페이지 *크기*가 충분한지 여부는 행의 바이트 합계 또는 컬럼 수에 의해 결정됩니다. 853 페이지의 『행 *크기*』에서 좀더 자세한 정보를 참조하십시오.

PARTITIONING KEY (*column-name,...*)

테이블의 데이터가 파티션된 경우에 사용되는 파티션 키를 지정합니다. 각 *컬럼 이름*은 테이블의 컬럼을 식별하고 같은 컬럼은 두 번 이상 식별되어서는 안됩니다.

이 절이 지정되지 않고 이 테이블이 여러 파티션 노드 그룹에 상주할 경우, 파티션 키는 선언된 임시 테이블의 첫번째 컬럼으로 정의됩니다.

선언된 임시 테이블이 단일 파티션 노드 그룹에 정의된 테이블 공간에 있을 경우, 컬럼의 모든 콜렉션은 파티션 키를 정의하는 데 사용할 수 있습니다. 이 매개변수를 지정하지 않으면, 파티션 키가 작성되지 않습니다.

파티션 키는 갱신할 수 없음을 주의하십시오(SQLSTATE 42997).

USING HASHING

데이터 분배를 위한 파티션 방법으로 해싱 함수의 사용을 지정합니다. 이것이 유일하게 지원되는 파티션 방법입니다.

주

- **선언된 전역 임시 테이블 참조:** 선언된 전역 임시 테이블의 설명은 DB2 카탈로그(SYSCAT.TABLES)에 나타나지 않으므로, 영속적이 아니고 데이터베이스 연결을 통해 공유 가능하지 않습니다. 이는 선언된 전역 임시 테이블 *table-name*을 정의하는 각 세션이 고유한 선언된 전역 임시 테이블 설명을 가짐을 의미합니다.

SQL문(DECLARE GLOBAL TEMPORARY TABLE 문이 아닌 다른)에서 선언된 전역 임시 테이블을 참조하기 위해서는 스키마 이름 SESSION으로 이 테이블을 명시적 또는 내재적 규정해야 합니다. *table-name*이 SESSION으로 규정되지 않은 경우에는 참조 해석시 선언된 전역 임시 테이블이 고려되지 않습니다.

이름으로 전역 임시 테이블을 선언하지 않은 연결에서 SESSION.*table-name*에 대한 참조는 카탈로그의 영속 오브젝트로부터 분석하려고 시도합니다. 이와 같은 오브젝트가 없으면, 오류가 발생합니다(SQLSTATE 42704).

- SESSION에 의해 내재적으로 또는 명시적으로 규정된 테이블을 참조하는 정적 SQL문이 있는 패키지를 바인딩하는 경우, 이러한 명령문이 정적으로 바인드되지 않습니다. 이러한 명령문은 호출될 때 패키지 바인딩시 선택된 VALIDATE 옵션에 관계없이 증분식으로 바인드됩니다. 런타임시 각 테이블 참조는 선언된 임시 테이블이나 영구 테이블로 해석됩니다. 둘다 없으면 오류가 발생합니다 (SQLSTATE 42704).
- **특권:** 선언된 전역 임시 테이블이 정의된 경우, 이 테이블의 정의자에게 테이블 삭제 능력을 비롯하여 테이블에 대한 모든 테이블 특권이 부여됩니다. 추가로 이러한 특권은 공용에 부여됩니다.[89] 이를 통해 세션의 모든 SQL문은 이미 해당 세션에 정의되어 있는 선언된 전역 임시 테이블을 참조할 수 있습니다.
- **인스턴스 작성 및 종료:** 아래 설명에서 P는 세션이고 T는 세션 P에서 선언된 전역 임시 테이블입니다.
  - 빈 인스턴스 T가 P에서 실행되는 DECLARE GLOBAL TEMPORARY TABLE 문의 결과로 작성됩니다.
  - P의 SQL문은 T로의 참조를 작성할 수 있습니다. P에서 T에 대한 모든 참조는 T의 동일한 인스턴스를 참조합니다.
  - DECLARE GLOBAL TEMPORARY TABLE 문이 SQL 프로시듀어 복합 텍스트(BEGIN 및 END로 정의)내에 지정되는 경우, 선언된 전역 임시 테이블의 범위는 복합 텍스트 명령문일 뿐만 아니라 연결 명령문이기도 하며 이 테이블은 복합 텍스트 명령문의 외부에 알려져 있습니다. 테이블은 복합 텍스트 명령문 종료시 내재적으로 삭제되지 않습니다. 선언된 전역 임시

89. GRANT 옵션으로 부여된 특권이 없고 카탈로그 테이블에 특권이 나타나지 않습니다.

테이블은 테이블이 명시적으로 삭제되지 않은한 해당 세션의 다른 복합 텍스트 명령문에 같은 이름으로 여러 번 정의할 수 없습니다.

- ON COMMIT DELETE ROWS 절이 내재적으로 또는 명시적으로 지정되었다고 가정할 때, 확약 조작이 P에서 작업 단위(UOW)를 종료하고 T에 종속적인 P에 열려 있는 WITH HOLD 커서가 없다면 확약에 DELETE FROM SESSION.T 조작이 포함됩니다.
- 구간 복원 조작이 P에서 작업 단위 또는 세이브포인트를 종료하고 작업 단위나 세이브포인트에 SESSION.T에 대한 수정사항이 포함된다면 구간 복원에 DELETE from SESSION.T가 포함됩니다.

  구간 복원 조작이 P에서 작업 단위 또는 세이브포인트를 종료하고 작업 단위나 세이브포인트에 SESSION.T의 선언이 포함된다면 구간 복원에 DROP SESSION.T 조작이 포함됩니다.

  구간 복원 조작이 P에서 작업 단위 또는 세이브포인트를 종료하고 작업 단위나 세이브포인트에 선언된 임시 테이블 SESSION.T의 삭제가 포함된다면 구간 복원이 테이블 삭제를 실행 취소하나 해당 테이블은 비게 됩니다.
- T를 선언한 응용프로그램 프로세스가 종료되고 데이터베이스로부터 연결해제된 경우, T가 삭제되고 인스턴스 작성된 행이 없어집니다.
- T가 선언된 서버로의 연결이 종료된 경우, T가 삭제되고 인스턴스 작성된 행이 없어집니다.

- **선언된 전역 임시 테이블 사용의 제한사항:** 선언된 전역 임시 테이블은 다음을 수행할 수 없습니다.
  - ALTER, COMMENT, GRANT, LOCK, RENAME 또는 REVOKE문에서의 정의(SQLSTATE 42995).
  - CREATE ALIAS, CREATE FUNCTION (SQL 스칼라, 테이블 또는 행), CREATE INDEX, CREATE TRIGGER 또는 CREATE VIEW문에서의 참조(SQLSTATE 42995).
  - 참조 제한조건에 지정(SQLSTATE 42995).

DELETE

DELETE문은 테이블이나 뷰에서 행을 삭제합니다. 뷰에서 행을 삭제하면 뷰가 기초로 하는 테이블에서 행이 삭제됩니다.

이 명령문은 두 가지의 형식을 갖고 있습니다.

- *검색* DELETE 형식은 두 개 이상의 행을 삭제할 경우 사용됩니다(검색 조건에 의해 선택적으로 판별됨).
- *위치지정* DELETE 형식은 정확히 한 행만 삭제할 경우 사용됩니다(커서의 현재 위치에 의해 판별되는 대로).

호출

DELETE는 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문의 두 형식 중 하나를 실행하려면, 명령문은 권한 부여 ID에서 갖고 있는 특권에는 최소한 다음 중 하나가 포함되어야 합니다.

- 행이 삭제될 테이블이나 뷰에서의 DELETE 특권
- 행이 삭제될 테이블이나 뷰에서의 CONTROL 특권
- SYSADM 또는 DBADM 권한

검색된 DELETE문을 실행하기 위해서는, 명령문의 권한 부여 ID가 보유한 특권에는 부속 조회에서 참조하는 각 테이블이나 뷰에 대해 다음 중 적어도 하나가 포함되어야 합니다.

- SELECT 특권
- CONTROL 특권
- SYSADM 또는 DBADM 권한

패키지가 SQL92 규칙을 사용하여 사전 처리 컴파일되고 [90] DELETE의 검색 형식에 *검색 조건*의 테이블이나 뷰의 컬럼에 대한 참조가 포함되는 경우, 그 명령문의 권한 부여 ID가 갖는 특권에는 다음 중 적어도 하나는 포함되어야 합니다.

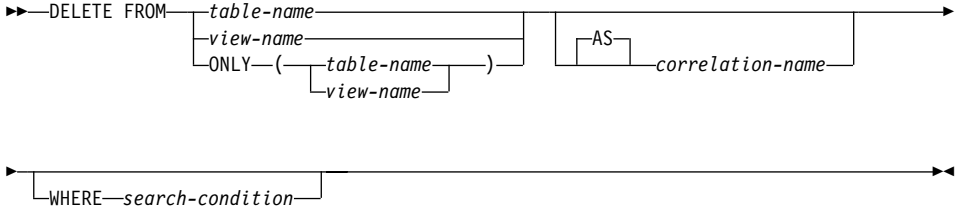
- SELECT 특권
- CONTROL 특권
- SYSADM 또는 DBADM 권한

지정된 테이블이나 뷰 앞에 ONLY 키워드가 올 때에는, 명령문의 권한 부여 ID에 의해 보유된 특권이 지정된 테이블이나 뷰의 모든 서브테이블이나 서브뷰에 대한 SELECT 특권도 포함해야 합니다.

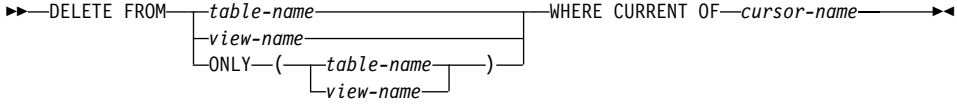
정적 DELETE문에 대해서는 Group 특권이 점검되지 않습니다.

구문

검색 **DELETE:**



위치 **DELETE:**



설명

FROM *table-name* 또는 *view-name*

행이 삭제될 테이블이나 뷰를 식별합니다. 이름은 카탈로그에 있는 테이블 또는 뷰를 식별해야 하나, 카탈로그 테이블, 카탈로그 뷰, 요약 테이블 또는 읽

90. 명령문을 처리하는 데 사용되는 패키지는 값이 SQL93E 또는 MIA인 LANGLEVEL 옵션을 사용하여 사전 처리 컴파일됩니다.

기 전용 뷰를 식별해서는 안 됩니다. (읽기 전용 뷰에 대해서는 931 페이지의 『CREATE VIEW』에서 참조하십시오.)

*table-name*이 입력된 테이블인 경우, 테이블 행 또는 적절한 서브테이블이 명령문에 의해 삭제될 수 있습니다.

*view-name*이 입력된 뷰라면, 뷰의 적절한 서브뷰의 기초가 되는 테이블이의 행이 명령문에 의해 삭제될 수도 있습니다. *view-name*이 입력된 테이블인 기초가 되는 테이블을 가진 정규 뷰라면, 입력된 테이블이나 적절한 서브의 행이 명령문에 의해 삭제될 수도 있습니다.

지정된 테이블의 컬럼들만이 WHERE절에서 참조될 수 있습니다. 위치지정된 DELETE의 경우, 연관된 커서가 ONLY를 사용하지 않고도 FROM절에서 테이블이나 뷰도 지정했어야 합니다.

FROM ONLY (*table-name***)**

입력된 테이블에 적용 가능한 ONLY 키워드는 명령문이 지정된 테이블 데이터에만 적용되고 해당 서브테이블 행은 삭제할 수 없도록 지정합니다. 위치지정된 DELETE인 경우, 연관된 커서가 ONLY를 사용하여 FROM절에 있는 테이블도 지정했을 것입니다. *table-name*이 입력된 테이블이 아닌 경우, ONLY 키워드는 명령문에 어떤 영향도 미치지 않습니다.

FROM ONLY (*view-name***)**

입력된 뷰에 적용 가능한 ONLY 키워드는 명령문이, 지정된 뷰 데이터에만 적용되고 해당 부속 뷰의 행은 삭제할 수 없도록 지정합니다. 위치지정된 DELETE인 경우, 연관된 커서가 ONLY를 사용하여 FROM절에 있는 뷰도 지정했을 것입니다. *view-name*이 입력된 뷰가 아닌 경우, ONLY 키워드는 명령문에 어떤 영향도 미치지 않습니다.

correlation-name

테이블이나 뷰를 지시하기 위해 검색 조건 내에서 사용될 수 있습니다. (상관 이름에 대한 설명을 보려면, 73 페이지의 『제3장 언어 요소』에서 참조하십시오.)

WHERE

삭제될 행을 선택하는 조건을 지정합니다. 절을 생략하거나, 검색 절을 지정하거나 커서를 명명할 수 있습니다. 절이 생략되면, 테이블이나 뷰의 모든 절이 삭제됩니다.

search-condition

236 페이지의 『검색 조건』에 설명된 검색 조건입니다. 부속 조회가 아닌, 검색 조건에서의 각 *column-name*은 테이블이나 뷰의 컬럼을 식별해야 합니다.

*search-condition*은 테이블이나 뷰의 각 행에 적용되고 삭제된 행은 *search-condition*이 참인 행들입니다.

검색 조건에 부속 조회가 있으면, 부속 조회는 *검색 조건*이 행에 적용될 때마다 실행되고, *검색 조건*을 적용할 때 사용되는 결과가 될 수 있습니다. 실제로, 상관된 참조가 없는 부속 조회는 한번 실행되는 반면, 상관된 참조를 갖는 부속 조회는 각 행에 한번씩 실행되어야 할 수도 있습니다. 부속 조회가 DELETE문의 오브젝트 테이블이나 CASCADE 또는 SET NULL의 삭제 규칙을 갖는 종속 테이블을 참조할 경우, 부속 조회는 행이 삭제되기 전에 완전히 평가됩니다.

CURRENT OF *cursor-name*

프로그램의 DECLARE CURSOR문에 정의되는 커서를 식별합니다. DECLARE CURSOR문 앞에는 DELETE문이 있어야 합니다.

명명된 테이블이나 뷰는 커서에 SELECT문의 FROM절에서도 명명되어야 하고, 커서의 결과 테이블은 읽기 전용이어야 합니다. (읽기 전용 테이블의 설명에 대해서는 952 페이지의 『DECLARE CURSOR』에서 참조하십시오.)

DELETE문이 실행될 때, 커서는 행에 위치되어야 합니다. 그 행은 한번 삭제됩니다. 삭제하고 나면, 커서는 결과 테이블의 다음 행 전에 위치됩니다. 다음 행이 없으면, 커서는 마지막 행 다음에 위치됩니다.

규칙

- 식별된 테이블이나 식별된 뷰의 기본 테이블이 상위 테이블일 경우, 삭제하기 위해 선택된 행에는 RESTRICT의 삭제 규칙과 관련하여 종속되는 것이 없어야 하며 DELETE는 RESTRICT의 삭제 규칙과 관련하여 종속 항목이 있는 종속 행에 연쇄될 수 없습니다.

  삭제 조작이 RESTRICT 삭제 규칙으로 금지되지 않는 경우, 선택된 행이 삭제됩니다. 선택된 행에 종속되는 행도 영향을 받습니다.

  – 또한, SET NULL의 삭제 규칙과의 관계에서 종속인 행에 대한 외부 키의 널(NULL) 값이 될 수 있는 컬럼은 널(NULL) 값으로 설정됩니다.
  – CASCADE의 삭제 규칙과의 관계에서 종속인 행도 삭제되고, 같은 규칙이 그 행에 적용됩니다.

  다른 참조 제한조건이 시행된 후 널(NULL) 값이 아닌 외부 키가 기존의 상위 행을 참조하도록 하기 위해 NO ACTION의 삭제 규칙이 점검됩니다.

주

- 여러 행으로 된 DELETE 실행시 오류가 발생하면, 데이터베이스가 변경되지 않습니다.
- 적절한 잠금이 이미 존재하지 않는 한, 성공적인 DELETE문 실행시 하나 이상의 독점 잠금이 확보됩니다. COMMIT 또는 ROLLBACK문을 발행하면 잠금 사항이 해제됩니다. 확약 및 구간 복원 조작으로 잠금이 해제될 때까지, 삭제 조작의 효과는 다음에 의해서만 감지될 수 있습니다.

  – 삭제를 수행한 응용프로그램 프로세스.
  – 분리 레벨 UR을 사용하는 다른 응용프로그램 프로세스.

  잠금은 다른 응용프로그램 프로세스가 테이블에서 수행되지 못하도록 합니다.
- 응용프로그램 프로세스에서 커서가 가리키는 행을 삭제할 경우, 그 커서들은 결과 테이블의 다음 행 전에 위치됩니다. 행 R 이전에 위치된 커서를 C라고 가정합니다(OPEN의 결과로, DELETE - C, DELETE - 일부 다른 커서 또는 검색된 DELETE). R이 유래되는 기본 테이블에 영향을 미치는 INSERT, UPDATE 및 DELETE 조작시, C를 참조하는 다음 FETCH 조작이 반드시 C

를 R에 위치시켜야 하는 것은 아닙니다. 예를 들어, 이 조작은 C를 R'에 위치시킬 수 있습니다. 여기서 R'은 현재 결과 테이블의 다음 행인 새 행입니다.

- SQLCA의 SQLERRD(3)는 명령문 실행 후에 오브젝트 테이블에서 삭제된 행 수를 보여줍니다. 여기에는 CASCADE 삭제 규칙의 결과로 삭제된 행은 포함하지 않습니다. SQLCA의 SQLERRD(5)는 참조 제한조건과 트리거된 명령문에 의해 영향을 받는 행 수를 보여줍니다. 여기에는 외부 키가 SET NULL 삭제 규칙의 결과로 NULL이 설정된 행과 CASCADE 삭제 규칙의 결과로 삭제된 행이 포함됩니다. 트리거된 명령문의 경우, 여기에는 삽입, 갱신 또는 삭제된 행 수도 포함됩니다. (SQLCA 설명에 관해서는 1261 페이지의 『부록B. SQL 통신(SQLCA)』에서 참조하십시오.)
- 검색 조건과 일치하는 모든 행과 기존의 참조 제한조건에 의해 요구되는 모든 작업을 삭제할 수 없는 오류가 발생하면, 테이블에서 어떠한 변경사항도 수행되지 않고 오류가 리턴됩니다.
- DATALINK 컬럼을 통해 현재 링크된 파일을 포함하는 삭제된 행의 경우, 파일 링크가 취소된 후 데이터 링크 컬럼 정의에 따라 복원되거나 삭제됩니다.

  값의 파일 서버가 더 이상 데이터베이스 서버로 등록되지 않는 경우 DATALINK 값을 삭제하려 하면 오류가 발생할 수 있습니다(SQLSTATE 55022).

  삭제시 사용불가능한 서버에 링크를 갖고 있는 행을 삭제하면 오류가 발생할 수도 있습니다(SQLSTATE 57050).

예

예 1: DEPARTMENT 테이블에서 부서(DEPTNO) 'D11'을 삭제합니다.

```
DELETE FROM DEPARTMENT
  WHERE DEPTNO = 'D11'
```

예 2: DEPARTMENT 테이블에서 모든 부서를 삭제합니다.(즉, 테이블을 비웁니다.)

```
DELETE FROM DEPARTMENT
```

DESCRIBE

DESCRIBE문은 준비된 명령문에 대한 정보를 얻습니다. 준비된 명령문에 대해 1087 페이지의 『PREPARE』에서 참조하십시오.

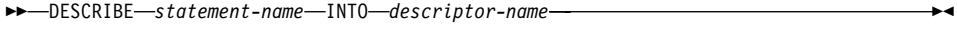
호출

이 명령문은 응용프로그램에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

필요사항 없음.

구문



설명

statement-name

정보가 필요한 명령문을 식별합니다. DESCRIBE문이 실행될 때, 이 설명은 준비된 명령문을 식별해야 합니다.

INTO *descriptor-name*

SQL 설명자 영역(SQLDA)을 식별합니다. SQLDA는 1267 페이지의 『부록 C. SQL 설명자 영역(SQLDA)』에 기술되어 있습니다. DESCRIBE문을 실행하기 전에 SQLDA에 있는 다음 변수가 설정되어 있어야 합니다.

SQLN SQLVAR로 표시되는 변수의 수를 나타냅니다. (SQLN은 SQLVAR 배열의 차원을 제공합니다.) SQLN이 제로 이상의 값으로 설정되어야 DESCRIBE문이 실행됩니다.

DESCRIBE 문이 실행될 때, 데이터베이스 관리 프로그램이 다음과 같이 값을 SQLDA의 변수에 지정합니다.

SQLDAID

첫번째 6바이트는 'SQLDA'로 설정되어 있습니다(즉, 5개 문자 뒤에 공백 문자로 구성).

SQLDA에 각 선택 목록 항목(또는, 결과 테이블의 *column*)별로 두 개의 SQLVAR 항목이 포함되는 경우, SQLDOUBLED라고 하는 7번째 바이트는 2로 설정됩니다. 이러한 기술은 LOB, 구별 유형, 구조화 유형 또는 참조 유형 결과 컬럼을 수용하는 데 사용됩니다. 그렇지 않은 경우, SQLDOUBLED는 공백 문자로 설정됩니다.

SQLDA에 DESCRIBE 응답 전체가 들어갈 만한 충분한 공간이 없는 경우, 이중 플래그가 공백으로 설정됩니다.

8번째 바이트는 공백 문자로 설정됩니다.

SQLDABC

SQLDA의 길이

SQLD 준비된 명령문이 SELECT인 경우, 그 결과 테이블의 컬럼 수. 그렇지 않은 경우, 0입니다.

SQLVAR

SQLD 값이 0이거나 SQLN 값보다 큰 경우, SQLVAR 어커런스에 아무 값도 지정되지 않습니다.

값이 *n*인 경우(여기서 *n*은 0보다 크나, SQLN값보다 작거나 같음을 의미), SQLVAR의 처음 *n* 어커런스에 값들이 지정되어, SQLVAR의 첫번째 어커런스에 결과 테이블의 첫번째 컬럼에 대한 설명이 포함되고, SQLVAR의 두 번째 어커런스에 결과 테이블의 두 번째 컬럼에 대한 설명이 포함되는 식으로 됩니다. 컬럼 설명은 SQLTYPE, SQLLEN, SQLNAME, SQLLONGLEN 및 SQLDATATYPE_NAME에 지정된 값들로 구성됩니다.

기본 SQLVAR

SQLTYPE

컬럼의 데이터 유형과 널(NULL) 값이 들어 있는지 여부를 보여주는 코드입니다.

SQLLEN

결과 컬럼의 데이터 유형에 따른 길이 값으로 SQLLEN은 LOB 데이터 유형에 대해 0입니다.

SQLNAME

파생된 컬럼이 단순한 컬럼 참조가 아닌 경우, sqlname에는 선택 목록 내에 있는 파생된 컬럼의 원래 위치를 나타내는 ASCII 숫자 문자형 상수값이 포함됩니다. 그렇지 않은 경우, sqlname에는 컬럼 이름이 포함됩니다.

보조 SQLVAR

이러한 변수는 SQLVAR 항목 수가 LOB, 구별 유형, 구조화 유형 또는 참조 유형 컬럼을 수용하도록 두 배가 되는 경우에만 사용됩니다.

SQLLONGLEN

BLOB, CLOB, DBCLOB 컬럼의 길이 속성.

SQLDATATYPE_NAME

사용자 정의 유형(구별 또는 구조화) 컬럼의 경우, 데이터베이스 관리 프로그램이 이를 완전한 사용자 정의 유형 이름으로 설정합니다. 참조 유형 컬럼의 경우, 데이터베이스 관리 프로그램이 이를 참조 목표 유형의 완전한 사용자 정의 유형 이름으로 설정합니다. 그렇지 않을 경우, 스키마 이름은 SYSIBM이고 유형 이름은 SYSCAT.DATATYPES 카탈로그 뷰의 TYPENAME 컬럼 이름입니다.

주

- DESCRIBE문이 실행되기 전에, SQLN값은 SQLDA에 SQLVAR 어커런스가 얼마나 많이 제공되어 있는지를 표시하도록 설정되고, SQLN 어커런스를 포함할 만한 충분한 저장영역이 할당되어야 합니다. 준비된 SELECT문의 결과 테이블 컬럼에 대한 설명을 보려면, SQLVAR 어커런스 수가 컬럼 수보다 적으면 안 됩니다.
- 큰 크기의 LOB가 예상되는 경우, 이러한 대형 오브젝트(LOB)를 조작하면 응용프로그램 메모리에 영향을 줄 수 있다는 점을 기억하십시오. 이러한 조건을 고려할 때, 위치 지정자나 파일 참조 변수를 고려하십시오. SQLLEN과 같은 다른 필드에 대한 해당 변경사항으로 SQL_TYP_xLOB의 SQLTYPE이 SQL_TYP_xLOB_LOCATOR 또는 SQL_TYP_xLOB_FILE로 변경되도록,

DESCRIBE문이 실행된 후 저장영역을 할당하기 전에 SQLDA를 수정하십시오. 그런 후, SQLTYPE에 기초한 저장영역을 할당한 후 계속하십시오.

SQLDA에서 위치 지정자와 파일 참조 변수를 사용하는 데 대해서는 *응용프로그램 개발 안내서*에서 좀더 자세한 정보를 참조하십시오.

- 확장 Unix 코드(EUC) 코드 페이지와 DBCS 코드 페이지 사이의 코드 페이지 변환으로 문자 길이가 확장 및 축소될 수 있습니다. 이러한 상황의 처리 방법에 대해서는 *응용프로그램 개발 안내서*에서 자세한 정보를 참조하십시오.
- 구조화 유형이 선택되지만 FROM SQL 변환은 지정되지 않은 경우(CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터를 사용하여 TRANSFORM GROUP이 지정되지 않았기 때문이거나(SQLSTATE 428EM) 명명된 그룹에 정의된 FROM SQL 변환 함수가 없기 때문(SQLSTATE 42744)), DESCRIBE는 오류를 리턴합니다.
- **SQLDA 할당:** SQLDA를 할당하는 가능한 방법 중 세 가지가 아래에 설명되어 있습니다.

  첫번째 기술: 응용프로그램이 처리해야 할 선택 목록을 수용할 수 있을 만큼 충분한 SQLVAR 발생시 SQLDA을 할당하십시오. 테이블에 LOB, 구별 유형, 구조화 유형 또는 참조 유형 컬럼이 포함되는 경우, SQLVAR 수는 최대 컬럼 수의 두 개이어야 합니다. 그렇지 않은 경우에는 SQLVAR 수가 최대 컬럼 수와 같아야 합니다. 할당을 마친 후 응용프로그램은 이 SQLDA를 반복적으로 사용할 수 있습니다.

  이 기술은, 특정 선택 목록에 대해 대부분의 저장영역이 사용되지 않은 경우에도, 할당 해제되지 않은 많은 양의 저장영역을 사용합니다.

  두 번째 기술: 처리되는 모든 선택 목록에 대해 다음 두 단계를 반복하십시오.

  1. SQLVAR 어커런스가 전혀 없는 SQLDA, 즉 SQLN이 제로인 SQLDA를 가지고 DESCRIBE문을 실행하십시오. SQLD에 대해 리턴된 값은 결과 테이블에 있는 컬럼 수입니다. 이것은 SQLVAR 어커런스에 필수적인 수이거나 필요한 수의 반값입니다. SQLVAR 항목이 없으므로, SQLSTATE 01005

를 가진 경고가 발행됩니다. 이러한 경고를 수반하는 SQLCODE가 +237, +238 또는 +239 중 하나와 같은 경우, SQLVAR 항목 수는 SQLD에 리턴되는 값의 두 배이어야 합니다. [91]

2. SQLVAR의 충분한 어커런스를 가지고 SQLDA를 할당하십시오. 그런 후, 이 새로운 SQLDA를 사용하여 다시 DESCRIBE문을 실행하십시오.

이 기술은 첫번째 기술보다 나은 저장영역 관리를 허용하나, DESCRIBE문의 수가 두 배가 됩니다.

세 번째 기술: 대부분 및 거의 모든 선택 목록을 처리하기에 충분하지만 합리적으로 *크기*가 작은 SQLDA를 할당하십시오. DESCRIBE를 실행한 후 SQLD 값을 검사하십시오. SQLVAR의 어커런스 수에 대한 SQLD값을 사용하여 필요한 경우, 보다 큰 SQLDA를 할당하십시오.

이 기술은 첫번째 두 기술 사이의 보완 기술에 해당합니다. 그 효과는 원래의 SQLDA에 대한 *크기*를 적절히 선택하는 것에 달려 있습니다.

예

C 프로그램에서, SQLVAR 어커런스가 전혀 없는 SALDA를 가지고 DESCRIBE 문을 실행합니다. SQLD이 0보다 큰 경우, 그 값을 사용하여 SQLVAR의 필요한 어커런스 수를 가진 SQLDA를 할당한 후, SQLDA를 사용하여 DESCRIBE문을 실행합니다.

```
EXEC SQL  BEGIN DECLARE SECTION;
  char  stmt1_str[200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;
... /* code to prompt user for a query, then to generate */
    /* a select-statement in the stmt1_str           */
EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
... /* code to set SQLN to zero and to allocate the SQLDA */
EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;
... /* code to check that SQLD is greater than zero, to set */
    /* SQLN to SQLD, then to re-allocate the SQLDA          */
EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;
```

91. 이러한 양수 SQLCODE의 리턴은 SQLWARN 바인드 옵션 설정이 YES(양수 SQLCODE 리턴)라고 가정합니다. SQLWARN이 NO로 설정된 경우에는 아직도 +238이 리턴되며, 이는 SQLVAR 항목 수가 SQLD에 리턴되는 값의 두 배이어야 함을 나타냅니다.

```
... /* code to prepare for the use of the SQLDA            */
    /* and allocate buffers to receive the data           */
EXEC SQL  OPEN DYN_CURSOR;
... /* loop to fetch rows from result table               */
EXEC SQL  FETCH DYN_CURSOR USING DESCRIPTOR :sqlda;
.
.
.
```

DISCONNECT

DISCONNECT 문은 사용 중인 작업 단위(UOW)가 없는 경우(즉, 확약이나 구간 복원 조작후) 하나 이상의 연결을 파괴합니다. [92]

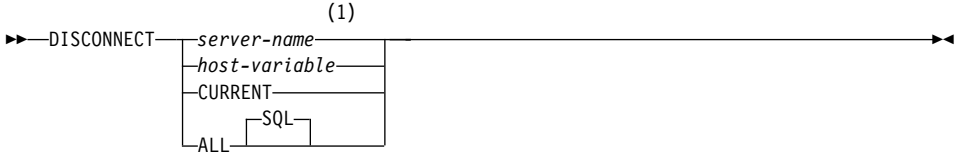
호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

필요사항 없음.

구문



주:

1 CURRENT 또는 ALL 응용프로그램 서버(AS)는 호스트 변수로만 식별됨에 유의하십시오.

설명

서버 이름 또는 *호스트 변수*

*서버 이름*이 들어 있는 *호스트 변수* 또는 지정된 *서버 이름*으로 응용프로그램 서버(AS)를 식별합니다.

92. 단일 연결이 DISCONNECT 문의 목표인 경우, 해당 연결은 사용 중인 작업 단위의 존재 여부에 관계없이 데이터베이스가 기존의 작업 단위에 참여하는 경우에만 파괴됩니다. 예를 들어 여러 다른 데이터베이스가 작업을 완료했으나 문의된 목표가 작업을 완료하지 않은 경우, 연결을 파괴하지 않고 연결해제할 수 있습니다.

*호스트 변수*가 지정되면, 길이가 8자 이하인 문자열 변수이어야 하며 표시기 변수를 포함해서는 안 됩니다. *호스트 변수* 내에 들어 있는 *서버 이름*은 왼쪽으로 정렬되어야 하고 따옴표로 구분해서는 안 됩니다.

*서버 이름*은 응용프로그램 서버(AS)를 식별하는 데이터베이스 별명임에 유의하십시오. 응용프로그램 리퀘스터의 지역 디렉토리에 나열되어야 합니다.

지정된 데이터베이스 별명 또는 호스트 변수에 들어 있는 데이터베이스 별명은 응용프로그램 프로세스의 기존 연결을 식별해야 합니다. 데이터베이스 별명이 기존의 연결을 식별하지 않은 경우, 오류(SQLSTATE 08003)가 발생합니다.

CURRENT

응용프로그램 프로세스의 현재 연결을 식별합니다. 응용프로그램 프로세스는 연결된 상태에 있어야 합니다. 그 외에는 오류(SQLSTATE 08003)가 발생합니다.

ALL

응용프로그램 프로세스의 모든 기존의 연결이 끊어지려 함을 나타냅니다. 명령문이 실행될 때 아무 것도 연결되어 있지 않으면 오류나 경고가 나타나지 않습니다. 선택적 키워드 SQL이 RELEASE문의 구문에 일치하도록 포함됩니다.

규칙

- 일반적으로, DISCONNECT문은 작업 단위(UOW) 내에서는 실행할 수 없습니다. 실행을 시도하면, 오류(SQLSTATE 25000)가 발생합니다. 이 규칙은 단일 연결이 끊어지도록 지정되고, 데이터베이스가 기존의 작업 단위(UOW)에 참여하지 않는 경우는 예외입니다. 이 경우, DISCONNECT문이 발행되었을 때 사용중인 작업 단위(UOW)가 있는 지는 문제가 되지 않습니다.
- DISCONNECT문은 트랜잭션 프로세싱 모니터 환경(SQLSTATE 25000)에서 실행될 수 없습니다. 이는 SYNCPOINT 사전 처리 컴파일러 옵션이 TWOPHASE로 설정되었을 때 사용됩니다.

주

- DISCONNECT문이 성공적으로 수행되면 식별된 각 연결은 끊어집니다.

DISCONNECT문이 성공적으로 수행되지 않으면, 응용프로그램 프로세스의 연결 상태 및 해당 연결 상태는 변경되지 않습니다.

- 현재 연결을 끊는 데 DISCONNECT가 사용되며, 다음 실행되는 SQL문은 CONNECT 또는 SET CONNECTION이어야 합니다.
- 유형 1 CONNECT 의미는 DISCONNECT 사용을 배제하지 않습니다. 그러나, DISCONNECT CURRENT와 DISCONNECT ALL을 사용할 수 있는 경우에도 CONNECT RESET문이 수행하는 것과 같은 확약 조작에서의 결과는 발생하지 않습니다.

  서버 이름 또는 *호스트 변수*가 DISCONNECT문에 지정되는 경우, 유형 1 CONNECT는 한 번에 하나의 연결만 지원하므로, 현재의 연결을 식별해야 합니다. 일반적으로, "규칙"에 언급된 예외를 갖는 작업 단위(UOW) 내에서는 DISCONNECT가 실패합니다.
- 자원은 원격 연결을 이루고 유지보수하는 데 필요합니다. 따라서, 재사용되지 않을 원격 연결은 가능한 한 빨리 해제되어야 합니다.
- 연결 옵션이 유효하므로 확약 조작시 연결이 끊어질 수도 있습니다. 연결 옵션은 AUTOMATIC, CONDITIONAL 또는 EXPLICIT가 될 수 있으며, 런타임 사전 처리 컴파일러 옵션으로 또는 SET CLIENT API를 통해 설정될 수 있습니다. DISCONNECT 옵션 스펙에 대한 45 페이지의 『분산 작업 단위(DUOW) 의미 정의 옵션』에서 자세한 정보를 참조하십시오.

예

예 1: 응용프로그램에는 더 이상 IBMSTHDB로의 SQL 연결이 필요하지 않습니다. 다음 명령문은 연결을 끊기 위해, 확약 또는 구간 복원 조작 이후에 실행되어야 합니다.

```
EXEC SQL DISCONNECT IBMSTHDB;
```

예 2: 응용프로그램에는 더 이상 현재의 연결이 필요하지 않습니다. 다음 명령문은 연결을 끊기 위해, 확약 또는 구간 복원 조작 이후에 실행되어야 합니다.

```
EXEC SQL DISCONNECT CURRENT;
```

예 3: 응용프로그램에서 더 이상 기존의 연결을 필요로 하지 않습니다. 다음 명령문은 모든 연결을 끊기 위해 확약 또는 구간 복원 조작 후에 실행되어야 합니다.

```
EXEC SQL DISCONNECT ALL;
```

DROP

DROP문은 오브젝트를 삭제합니다. 그 오브젝트에 직접 또는 간접으로 의존하고 있는 모든 오브젝트는 삭제되거나 작동되지 않게 됩니다.(887 페이지의 『작동 불능 트리거』와 942 페이지의 『작동 불능 뷰』에서 자세한 내용을 참조하십시오.) 오브젝트가 삭제될 때마다, 카탈로그에서 그 내용을 삭제하고 이 오브젝트를 참조하는 모든 패키지는 유효하지 않게 됩니다.

호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

두 부분으로 된 이름을 허용하는 오브젝트 삭제시 DROP문의 권한 부여 ID가 보유해야 하는 특권은 다음 중 하나를 포함해야 하며, 그렇지 않을 경우 오류가 발생합니다(SQLSTATE 42501).

- SYSADM 또는 DBADM 권한
- 오브젝트의 스키마에 대한 DROPIN 특권
- 오브젝트에 대한 카탈로그 뷰의 DEFINER 컬럼에 레코드된 오브젝트의 정의자
- 오브젝트상의 CONTROL 특권(색인, 색인 스펙, 별명, 패키지, 테이블 및 뷰에만 해당)
- 카탈로그 뷰 SYSCAT.DATATYPES의 DEFINER 컬럼에 기록된 사용자 정의 유형의 정의자(사용자 정의 유형과 연관된 메소드 삭제를 삭제하는 경우에만 적용 가능)

테이블이나 뷰 계층을 삭제할 때 DROP문의 권한 부여 ID는 계층에 있는 테이블이나 뷰 각각에 대해 위의 특권 중 하나를 보유해야 합니다.

스키마를 삭제할 경우 DROP문의 권한 부여 ID는 SYSCAT.SCHEMATA의 OWNER 컬럼에 기록된 스키마 소유자이거나 SYSADM 또는 DBADM 권한을 가져야 합니다.

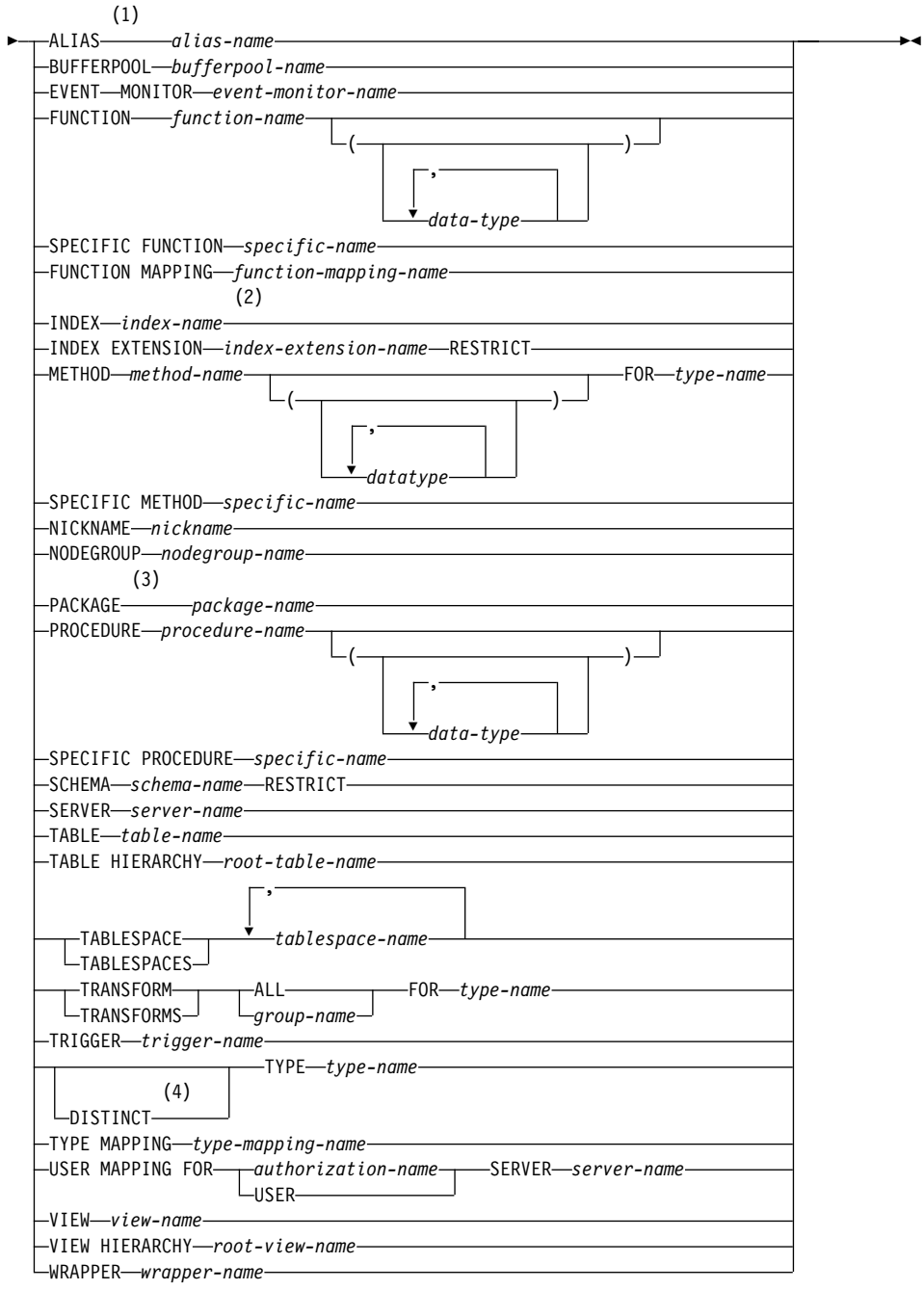
버퍼 풀, 노드 그룹 또는 테이블 공간을 삭제할 경우 DROP문의 권한 부여 ID는 SYSADM 또는 SYSCTRL 권한을 가져야 합니다.

이벤트 모니터, 서버 정의, 데이터 유형 맵핑, 함수 맵핑 또는 랩퍼를 삭제할 때에는 DROP문의 권한 부여 ID가 SYSADM 또는 DBADM 권한을 가져야 합니다.

사용자 맵핑을 삭제할 때에는, 권한 부여 ID가 맵핑 내의 연합 데이터베이스 권한 부여 이름과 다를 경우 DROP문의 이 권한 부여 ID에 SYSADM 또는 DBADM 권한이 있어야 합니다. 그렇지 않으면, 권한 부여 ID가 권한 부여 이름과 일치하는 경우 어떠한 권한이나 특권도 필요하지 않습니다.

변환 삭제시 DROP문의 권한 부여 ID는 SYSADM 또는 DBADM 권한을 보유하거나 *type-name*의 DEFINER이어야 합니다.

구문


(1)
ALIAS alias-name
BUFFERPOOL bufferpool-name
EVENT MONITOR event-monitor-name
FUNCTION function-name
(, data-type)
SPECIFIC FUNCTION specific-name
FUNCTION MAPPING function-mapping-name
(2)
INDEX index-name
INDEX EXTENSION index-extension-name RESTRICT
METHOD method-name
(, datatype)
FOR type-name
SPECIFIC METHOD specific-name
NICKNAME nickname
NODEGROUP nodegroup-name
(3)
PACKAGE package-name
PROCEDURE procedure-name
(, data-type)
SPECIFIC PROCEDURE specific-name
SCHEMA schema-name RESTRICT
SERVER server-name
TABLE table-name
TABLE HIERARCHY root-table-name
TABLESPACE
TABLESPACES
, tablespace-name
TRANSFORM
TRANSFORMS
ALL
group-name
FOR type-name
TRIGGER trigger-name
TYPE type-name
DISTINCT
(4)
TYPE MAPPING type-mapping-name
USER MAPPING FOR
authorization-name
USER
SERVER server-name
VIEW view-name
VIEW HIERARCHY root-view-name
WRAPPER wrapper-name

주:

1 ALIAS에 대한 동의어로 SYNONYM을 사용할 수 있습니다.

2 *Index-name*은 색인 또는 색인 스펙의 이름일 수 있습니다.

3 PACKAGE에 대한 동의어로 PROGRAM을 사용할 수 있습니다.

4 사용자 정의 유형을 삭제할 때 DATA도 사용할 수 있습니다.

설명

ALIAS *alias-name*

삭제될 별명을 식별합니다. *alias-name*은 카탈로그에서 설명되는 별명을 식별해야 합니다(SQLSTATE 42704). 지정된 별명은 삭제됩니다.

별명을 참조하는 모든 테이블, 뷰 및 트리거[93]는 실행 불능 상태가 됩니다.

BUFFERPOOL *bufferpool-name*

삭제될 버퍼 풀을 식별합니다. *bufferpool-name*은 카탈로그에 기술된 버퍼 풀을 식별해야 합니다(SQLSTATE 42704). 버퍼 풀에 지정된 테이블 공간이 있을 수 없습니다(SQLSTATE 42893). IBMDEFAULTBP 버퍼 풀은 삭제할 수 없습니다(SQLSTATE 42832). 데이터베이스가 정지될 때까지 버퍼 풀 저장영역은 릴리스되지 않습니다.

EVENT MONITOR *event-monitor-name*

삭제될 이벤트 모니터를 식별합니다. *event-monitor-name*은 카탈로그에 기술되어 있는 이벤트 모니터(SQLSTATE 42704)를 식별해야 합니다.

식별된 이벤트 모니터가 ON인 경우, 오류(SQLSTATE 55034)가 발생합니다. 그렇지 않은 경우, 이벤트 모니터는 삭제됩니다.

이벤트 모니터가 삭제될 때 이벤트 모니터의 목표 경로에 이벤트 파일이 있는 경우, 이벤트 파일은 삭제되지 않습니다. 그러나, 동일한 목표 경로를 지정하는 새로운 이벤트 모니터가 작성되는 경우, 이벤트 파일은 삭제됩니다.

FUNCTION

삭제될 사용자 정의 함수(완전한 함수 또는 함수 템플리트)의 인스턴스를 식

93. 여기에는 CREATE TRIGGER문의 ON절에서 참조된 테이블과 트리거 SQL문 내에서 참조된 모든 테이블이 포함됩니다.

별합니다. 지정된 함수 인스턴스는 카탈로그에 기술되어 있는 사용자 정의 함수여야 합니다. CREATE DISTINCT TYPE문에 의해 내재적으로 생성되는 함수는 삭제할 수 없습니다.

함수 인스턴스를 식별하는 데 사용할 수 있는 방법이 여러 가지 있습니다.

FUNCTION *function-name*

특정 함수를 식별하고, *function-name*이 있는 함수 인스턴스가 하나인 경우에만 유효합니다. 그러므로 식별되는 함수는 이에 대해 정의된 임의 수의 매개변수를 가질 수 있습니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 사전컴파일/바인드 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 명명되거나 암시적인 스키마에 이 이름을 가진 함수가 없으면, 오류(SQLSTATE 42704)가 발생합니다. 명명된 또는 암시적인 스키마에 하나 이상의 특정 함수 인스턴스가 있는 경우, 오류(SQLSTATE 42854)가 발생합니다.

FUNCTION *function-name* **(***data-type***,...)**

삭제될 함수를 고유하게 식별하는 함수 시그니처를 제공합니다. 함수 선택 알고리즘은 사용하지 않습니다.

function-name

삭제될 함수의 함수 이름을 부여합니다. 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 사전컴파일/바인드 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

(data-type,...)

해당 위치에 있는 CREATE FUNCTION문에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형의 수, 데이터 유형의 논리적 병합이 사용되어 삭제될 특정 함수 인스턴스를 식별합니다.

*data-type*이 규정되어 있지 않으면, SQL 경로의 스키마를 검색하여 유형 이름을 해석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대한 정밀도 또는 스케일, 길이를 지정하는 것은 필요하지 않습니다. 대신, 데이터 유형 일치를 찾을 때 이들 속성이 무시됨을 표시하기 위해 빈 괄호를 쓸 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용될 수 없습니다(SQLSTATE 42601).

그러나, 길이, 정밀도 또는 스케일이 코드화된 경우, 그 값은 CREATE PROCEDURE문에 지정된 것과 정확히 일치해야 합니다.

0 <n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치(Match)는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

지정된 이름을 지닌 함수가 명명된 스키마 또는 암시된 스키마에 없는 경우, 오류가 발생합니다(SQLSTATE 42883).

SPECIFIC FUNCTION *specific-name*

함수 생성시에 지정되거나 기본값으로 되는 특정 이름을 사용하여, 삭제될 특정 사용자 정의 함수를 식별합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 사전컴파일/바인드 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. *specific-name*은 명명된 또는 암시된 스키마에서 특정 함수 인스턴스를 식별합니다. 그렇지 않은 경우, 오류가 발생합니다(SQLSTATE 42704).

SYSIBM 스키마 또는 SYSFUN 스키마(SQLSTATE 42832)에 있는 함수를 삭제할 수 없습니다.

다른 오브젝트는 함수에 종속적일 수 있습니다. 그러한 모든 종속 오브젝트는 함수가 삭제되기 전에 실행 불능으로 표시된 패키지를 제외하고는 삭제되어야 합니다. 그러한 종속 함수를 지닌 함수를 삭제하려 하면 오류(SQLSTATE 42893)가 발생합니다. 종속 함수 목록에 대해서는 1003 페이지를 참조하십시오.

함수가 삭제 가능한 경우, 삭제됩니다.

삭제 중인 특정 함수에 종속적인 패키지는 실행 불능으로 표시됩니다. 그러한 패키지는 암시적으로 리바인드되지 않습니다. BIND 또는 REBIND 명령을 사용하여 리바인드되거나, PREP 명령을 사용하여 다시 준비되어야 합니다. 이 명령에 대해서는 *Command Reference*에서 참조하십시오.

FUNCTION MAPPING *function-mapping-name*

삭제될 함수 맵핑을 식별합니다. *function-mapping-name*은 카탈로그에서 설명되는 사용자 정의 함수(UDF) 맵핑을 식별해야 합니다(SQLSTATE 42704). 함수 맵핑이 데이터베이스에서 삭제됩니다.

기본 함수 맵핑은 삭제될 수 없습니다. 그러나, 사용안함으로 될 수는 있습니다. 예를 들어, 735 페이지의 『CREATE FUNCTION MAPPING』의 예 3을 보십시오.

삭제된 함수 맵핑에 종속성을 가지는 패키지는 무효화됩니다.

INDEX *index-name*

삭제될 색인이나 색인 스펙을 식별합니다. *index-name*은 카탈로그에 기술되어 있는 색인 또는 색인 스펙을 식별해야 합니다(SQLSTATE 42704). 이 색인은 시스템이 기본 키 또는 고유 제한조건이나 복제된 요약 테이블용으로 필요로 하는 색인이어서는 안 됩니다(SQLSTATE 42917). 지정된 색인이나 색인 스펙이 삭제됩니다.

삭제된 색인이나 색인 스펙에 종속성을 가지는 패키지는 무효화됩니다.

INDEX EXTENSION *index-extension-name* **RESTRICT**

삭제될 확장 색인을 식별합니다. *index-extension-name*은 카탈로그에 기술되어 있는 색인 확장을 식별해야 합니다(SQLSTATE 42704). RESTRICT 키워드는 이 색인 확장 정의에 따르는 색인을 정의할 수 없다는 규칙을 시행합니다(SQLSTATE 42893).

METHOD

삭제될 메소드 본문을 식별합니다. 지정된 메소드 내용은 카탈로그에 기술된 메소드이어야 합니다(SQLSTATE 42704). 내재적으로 CREATE TYPE에 의해 생성된 메소드 내용은 삭제할 수 없습니다.

DROP METHOD는 메소드 내용을 삭제하나, 메소드 스펙(시그니처)은 주제 유형 정의의 한 부분으로 남습니다. 메소드 내용 삭제후, ALTER TYPE DROP METHOD로 메소드 스펙을 주제 유형 정의에서 제거할 수 있습니다.

삭제될 메소드 내용을 식별하는 데 사용할 수 있는 방법은 여러가지가 있습니다.

METHOD *method-name*

삭제된 특정 메소드를 식별하며, 이름은 *method-name*이고 주제 유형은 *type-name*인 메소드 인스턴스가 정확히 하나 있는 경우에만 유효합니다. 즉, 식별된 메소드는 임의 수의 매개변수를 가질 수 있습니다. *type-name* 유형에 대해 이 이름의 메소드가 없는 경우에는 오류가 발생합니다 (SQLSTATE 42704). 명명된 데이터 유형에 대해 메소드의 특정 인스턴스가 둘 이상 있는 경우에는 오류가 발생합니다(SQLSTATE 42854).

METHOD *method-name* **(***data-type***,...)**

삭제될 메소드를 고유하게 식별하는 메소드 시그니처를 제공합니다. 메소드 선택 알고리즘은 사용하지 않습니다.

method-name

지정된 유형에 대해 삭제될 메소드의 메소드 이름. 이 이름은 규정화되지 않은 식별자이어야 합니다.

(*data-type***, ...)**

CREATE TYPE 또는 ALTER TYPE문의 메소드 스펙 중 해당 위치에 지정된 데이터 유형과 일치해야 합니다. 데이터 유형 수와 데이터 유형의 논리적 병합을 사용하여 삭제될 특정 메소드 인스턴스를 식별합니다.

데이터 유형이 규정되어 있지 않으면, SQL 경로의 스키마를 검색하여 유형 이름을 해석합니다.

매개변수화된 데이터 유형에 대한 정밀도 또는 스케일, 길이를 지정하는 것은 필요하지 않습니다. 대신, 데이터 유형 일치를 찾을 때 이들 속성이 무시됨을 표시하기 위해 빈 괄호를 쓸 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용될 수 없습니다(SQLSTATE 42601).

그러나 길이, 정밀도 또는 스케일이 코드화된 경우, 그 값은 CREATE TYPE문에 지정된 것과 정확히 일치해야 합니다.

0 <n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치(Matching)는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

명명된 데이터 유형에 대해 지정된 시그니처를 갖는 메소드가 없는 경우, 오류가 발생합니다(SQLSTATE 42883).

FOR *type-name*

지정된 메소드가 삭제될 유형을 명명합니다. 이 이름은 카탈로그에 이미 기술된 유형을 식별해야 합니다(SQLSTATE 42704). 동적 SQL문에서 CURRENT SCHEMA 특수 레지스터는 규정되지 않은 유형 이름용 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 사전 처리 컴파일/바인드 옵션은 내재적으로 규정되지 않은 유형 이름용 규정자를 지정합니다.

SPECIFIC METHOD *specific-name*

CREATE TYPE 또는 ALTER TYPE 수행시 지정되었거나 기본값으로 설정된 이름을 사용하여 삭제될 특정 메소드를 식별합니다. 고유 이름이 규정되지 않은 경우, CURRENT SCHEMA 특수 레지스터는 동적 SQL에서 규정되지 않은 고유 이름용 규정자로 사용됩니다. 정적 SQL문에서 QUALIFIER 사전 처리 컴파일/바인드 옵션은 내재적으로 규정되지 않은 고유 이름용 규정자를 지정합니다. specific-name은 메소드를 식별합니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 42704).

다른 오브젝트는 메소드에 종속적일 수 있습니다. 그러한 모든 종속 오브젝트는 메소드가 삭제되기 전에 실행 불능으로 표시된 패키지를 제외하고는 삭제되어야 합니다. 그러한 종속 함수를 지닌 메소드를 삭제하려 하면 오류가 발생합니다(SQLSTATE 42893).

메소드를 삭제할 수 있는 경우 메소드는 삭제됩니다.

삭제 중인 특정 메소드에 종속적인 패키지는 작동 불능으로 표시됩니다. 그러한 패키지는 내재적으로 다시 바인드됩니다. 이 패키지는 BIND 또는 REBIND

명령을 사용하여 리바인드하거나 PREP 명령을 사용하여 다시 준비해야 합니다. 이러한 명령에 대해서는 *Command Reference*를 참조하십시오.

NICKNAME *별명*

삭제될 별명을 식별합니다. 별명은 카탈로그에서 나열되어야 합니다(SQLSTATE 42704). 별명이 데이터베이스에서 삭제됩니다.

별명과 연관된 컬럼 및 색인에 대한 모든 정보가 카탈로그에서 삭제됩니다. 별명에 종속되는 모든 색인 스펙이 삭제됩니다. 별명에 종속하는 모든 뷰는 작동 불능으로 표시됩니다. 삭제된 색인 스펙에 종속하는 모든 패키지가 무효화되거나 작동 불능 뷰가 무효화됩니다. 별명이 참조하는 데이터 소스 테이블은 영향받지 않습니다.

NODEGROUP *nodegroup-name*

삭제될 노드 그룹을 식별합니다. *nodegroup-name*은 카탈로그에 서술된 노드 그룹을 식별해야 합니다(SQLSTATE 42704). 이 이름은 한 부분의 이름입니다.

노드 그룹을 삭제하면 노드 그룹에 정의된 모든 테이블 공간이 삭제됩니다. 테이블 공간에 있는 테이블에 종속되는 기존의 모든 데이터베이스 오브젝트(예. 패키지, 참조 제한조건 등)가 삭제되거나 무효가 되며(해당되는 경우), 종속 뷰 및 트리거를 사용할 수 없게 됩니다.

시스템 정의 노드 그룹은 삭제할 수 없습니다(SQLSTATE 42832).

DROP NODEGROUP이 현재 데이터 재분배에 영향을 받은 노드그룹에 대해 발행될 경우, DROP NODEGROUP 연산은 실패하게 되고 오류를 리턴합니다(SQLSTATE 55038). 그러나 부분적으로 재분배된 노드 그룹을 삭제할 수 있습니다. REDISTRIBUTE NODEGROUP 명령이 완료에 실행되지 못하면 노드 그룹은 부분적으로 재분배됩니다. 이는 오류로 인터럽트되거나 응용프로그램이 모든 명령을 강제 중단하는 경우 발생할 수 있습니다.[94]

94. 부분적으로 재분배되는 노드 그룹의 경우, SYSCAT.NODEGROUPS 카탈로그의 REBALANCE_PMAP_ID는 -1이 아닙니다.

PACKAGE *package-name*

삭제될 패키지를 식별합니다. *package-name*은 카탈로그에 서술된 패키지를 식별해야 합니다(SQLSTATE 42704). 지정된 패키지가 삭제됩니다. 패키지상의 모든 특권 또한 삭제됩니다.

PROCEDURE

삭제될 저장 프로시듀어 인스턴스를 식별합니다. 지정된 프로시듀어 인스턴스는 카탈로그에 기술되어 있는 저장 프로시듀어야 합니다.

프로시듀어 인스턴스를 식별하는 데 사용할 수 있는 방법이 여러 가지 있습니다.

PROCEDURE *procedure-name*

특정 프로시듀어를 식별하고, 스키마 내에 *프로시듀어 이름*이 있는 프로시듀어 인스턴스가 하나인 경우에만 유효합니다. 그러므로 식별되는 프로시듀어에 대해서는 수에 관계없이 매개변수를 정의할 수 있습니다. 명명되거나 내재된 스키마에 이 이름의 프로시듀어가 존재하지 않는 경우, 오류(SQLSTATE 42704)가 발생합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 명명된 스키마나 내재된 스키마에 프로시듀어의 특정 인스턴스가 하나 이상 있을 경우, 오류(SQLSTATE 42854)가 발생합니다.

PROCEDURE *procedure-name* **(***data-type***,...)**

삭제될 프로시듀어를 고유하게 식별하는 프로시듀어 시그니처를 제공합니다. 프로시듀어 선택 알고리즘은 사용하지 않습니다.

procedure-name

삭제될 프로시듀어 이름을 제공합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 사전컴파일/바인드 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다.

(data-type,...)

해당 위치에 있는 CREATE PROCEDURE문에 지정된 데이터 유형

과 일치해야 합니다. 데이터 유형의 수, 데이터 유형의 논리적 병합을 사용하여 삭제될 특정 프로시듀어 인스턴스를 식별합니다.

*data-type*이 규정되어 있지 않으면, SQL 경로의 스키마를 검색하여 유형 이름을 해석합니다. REFERENCE 유형에 대해 지정된 데이터 유형 이름에도 적용됩니다.

매개변수화된 데이터 유형에 대한 정밀도 또는 스케일, 길이를 지정하는 것은 필요하지 않습니다. 대신, 데이터 유형 일치를 찾을 때 이들 속성이 무시됨을 표시하기 위해 빈 괄호를 쓸 수 있습니다.

매개변수 값이 다른 데이터 유형(REAL 또는 DOUBLE)을 나타내기 때문에 FLOAT()는 사용될 수 없습니다(SQLSTATE 42601).

그러나, 길이, 정밀도 또는 스케일을 적어넣는 경우, 그 값은 CREATE FUNCTION문에 지정된 값과 정확히 일치해야 합니다.

0 <n<25는 REAL 유형이고 24<n<54는 DOUBLE 유형을 의미하기 때문에 FLOAT(n) 유형이 n에 대해 정의된 값과 일치할 필요는 없습니다. 일치는 유형이 REAL인지 또는 DOUBLE인지에 따라 발생합니다.

명명되거나 내재된 스키마에 지정된 시그니처가 있는 프로시듀어가 존재하지 않는 경우, 오류가 발생합니다(SQLSTATE 42883).

SPECIFIC PROCEDURE *specific-name*

프로시듀어 생성시에 지정되거나 기본값으로 되는 특정 이름을 사용하여, 삭제될 특정 저장 프로시듀어를 식별합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER precompile/bind 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. *specific-name*은 명명된 스키마 또는 내재된 스키마의 특정 프로시듀어 인스턴스를 식별해야 합니다. 그렇지 않으면, 오류(SQLSTATE 42704)가 발생합니다.

SCHEMA *schema-name* **RESTRICT**

삭제될 스키마를 식별합니다. *schema-name*은 카탈로그에 기술된 스키마를 식

별해야 합니다(SQLSTATE 42704). RESTRICT 키워드는 스키마가 데이터베이스에서 삭제되도록 지정된 스키마에 오브젝트를 정의할 수 없다는 규칙을 제시합니다(SQLSTATE 42893).

SERVER *server-name*

그 정의가 카탈로그에서 삭제될 데이터 소스를 식별합니다. *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다(SQLSTATE 42704). 데이터 소스의 정의가 삭제됩니다.

데이터 소스에 상주하는 테이블 및 뷰에 대한 모든 별명이 삭제됩니다. 이들 별명에 종속된 모든 색인 스펙이 삭제됩니다. 삭제된 서버 정의에 종속되는 모든 사용자 정의 함수(UDF) 맵핑, 사용자 정의 유형 맵핑 및 사용자 맵핑 역시 삭제됩니다. 삭제된 서버 정의, 함수 맵핑, 별명 및 색인 스펙에 종속되는 모든 패키지가 무효화됩니다.

TABLE *table-name*

삭제될 기본 테이블, 선언된 임시 테이블 또는 요약 테이블을 식별합니다. *table-name*은 카탈로그에 기술된 테이블을 식별해야 하며, 선언된 임시 테이블인 경우에는 *table-name*이 스키마 이름 SESSION으로 규정되고 응용프로그램에 존재해야 합니다(SQLSTATE 42704). 입력된 테이블의 서브테이블은 상위 테이블에 종속됩니다. 모든 서브테이블을 삭제해야 상위 테이블이 삭제될 수 있습니다(SQLSTATE 42893). 지정된 테이블이 데이터베이스에서 삭제됩니다.

테이블을 참조하는 모든 색인, 기본 키, 외부 키, 점검 제한조건 및 요약 테이블이 삭제됩니다. 테이블을 참조하는 모든 뷰 및 트리거[95]가 작동 불능으로 됩니다. 실행 불능으로 표시되거나 삭제된 오브젝트와 관련된 모든 패키지는 유효하지 않게 됩니다. 여기에는 계층에서 서브테이블 위에 있는 상위 테이블에 종속된 패키지가 포함됩니다. 삭제된 테이블이 참조 영역으로서 정의된 참조 컬럼의 영역이 해제됩니다.

패키지는 선언된 임시 테이블에 종속되지 않으므로 그러한 테이블이 삭제될 때 무효화되지 않습니다.

---

95. 여기에는 CREATE TRIGGER문의 ON절에서 참조된 테이블과 트리거된 SQL문 내에서 참조하는 모든 테이블 모두가 포함됩니다

DATALINK 컬럼을 통해 링크되는 모든 파일들의 링크가 해제됩니다. 언링크 조작은 비동기식으로 수행되므로 파일을 다른 조작에 즉시 사용할 수 없습니다.

테이블 계층에서 서브테이블이 삭제되면, 컬럼 수 및 행 크기 한계와 관련된 사항이라도 서브테이블과 연관된 컬럼에 더 이상 액세스할 수 없습니다. 서브테이블 삭제는 서브테이블의 모든 행을 수퍼테이블에서 삭제하는 데 영향을 미칩니다. 서브테이블 삭제 결과로 수퍼테이블에 정의된 트리거 또는 참조 무결성 제한조건이 활성화될 수 있습니다.

선언된 임시 테이블이 삭제되고 작업 단위 활성화 또는 세이브포인트에 앞서 작성되는 경우, 이 테이블은 기능적으로 삭제되고 응용프로그램이 테이블에 액세스할 수 없게 됩니다. 그러나 테이블은 아직 테이블 공간의 일부 공간을 차지하며 작업 단위가 확약되거나 세이브포인트가 종료될 때까지 사용자 임시 테이블 공간이 삭제되지 않도록 방지하고 사용자 임시 테이블 공간의 노드 그룹이 재분배되지 않도록 방지합니다. 선언된 임시 테이블을 삭제하면 DROP의 확약 또는 구간 복원 여부에 관계없이 테이블의 데이터가 파괴됩니다.

TABLE HIERARCHY *root-table-name*

삭제될 입력된 테이블 계층을 식별합니다. *root-table-name*은 입력된 테이블 계층에서 루트 테이블인 입력된 테이블을 식별해야 합니다(SQLSTATE 428DR). *root-table-name*에 의해 식별된 입력된 테이블과 모든 서브테이블이 데이터베이스에서 삭제됩니다.

삭제된 테이블을 참조하는 모든 색인, 요약 테이블, 기본 키, 외부 키 및 점검 제한조건이 삭제됩니다. 삭제된 테이블을 참조하는 모든 뷰 및 트리거가 작동 불능으로 됩니다. 실행 불능으로 표시되거나 삭제된 오브젝트와 관련된 모든 패키지는 유효하지 않게 됩니다. 삭제된 테이블 중 하나에 대한 영역으로 정의된 모든 참조 컬럼의 영역이 해제됩니다.

DATALINK 컬럼을 통해 링크되는 모든 파일들의 링크가 해제됩니다. 언링크 조작은 비동기식으로 수행되므로 파일을 다른 조작에 즉시 사용할 수 없습니다.

하나의 테이블을 삭제하는 것과는 달리, 테이블 계층을 삭제하면 계층에 있는 테이블의 삭제 트리거를 활성화시키지도 않고, 삭제된 행을 로그하지도 않습니다.

TABLESPACE 또는 **TABLESPACES** *tablespace-name*

삭제될 테이블 공간을 식별합니다. *tablespace-name*은 카탈로그에서 설명되는 테이블 공간을 식별해야 합니다(SQLSTATE 42704). 이 이름은 한 부분의 이름입니다.

삭제 중인 테이블 공간에 그 부분 중 최소한 하나를 저장하고 삭제되고 있지 않은 다른 테이블 공간에 그 부분 중 하나 이상이 있는 테이블이 있다면 테이블 공간이 삭제되지 않습니다(SQLSTATE 55024)(이러한 테이블을 먼저 삭제해야 합니다). 시스템 테이블 공간은 삭제할 수 없습니다(SQLSTATE 42832). SYSTEM TEMPORARY 테이블 공간은 데이터베이스에 임시 테이블 공간만 있는 경우에는 삭제할 수 없습니다(SQLSTATE 55026). 사용자 임시 테이블 공간은 그 안에 선언된 임시 테이블이 작성된 경우 삭제할 수 없습니다(SQLSTATE 55039). 선언된 임시 테이블이 삭제되었을지라도 사용자 임시 테이블 공간은 DROP TABLE을 포함하는 작업 단위가 확약될 때까지 여전히 사용 중인 것으로 간주됩니다.

테이블 공간을 삭제하면 테이블 공간에 정의된 모든 오브젝트가 삭제됩니다. 패키지, 참조 제한조건 등과 같이, 테이블 공간상에서 종속성이 있는 기존의 모든 데이터베이스 오브젝트는 삭제되거나 무효가 되며(해당되는 경우), 종속 뷰 및 트리거가 작동되지 않게 됩니다.

사용자가 작성하는 컨테이너는 삭제되지 않습니다. CREATE TABLESPACE에서 데이터베이스 관리 프로그램이 작성한 컨테이너 이름의 경로에 있는 디렉토리는 삭제됩니다. 데이터베이스 디렉토리 아래의 모든 컨테이너가 삭제됩니다. SMS 테이블 공간의 경우, 모든 연결이 해제되었거나 DEACTIVATE DATABASE 명령이 발행된 후에 삭제가 발생합니다.

TRANSFORM ALL FOR *type-name*

사용자 정의 데이터 유형 *type-name*에 대해 정의된 모든 변환 그룹이 삭제될 것임을 나타냅니다. 이러한 그룹에서 참조되는 변환 함수는 삭제되지 않습니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 사전 컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. *type-name*은 카탈로그에서 설명되는 사용자 정의 유형을 식별해야 합니다(SQLSTATE 42704).

*type-name*에 대해 정의된 변환이 없는 경우 오류가 발생합니다(SQLSTATE 42740).

DROP TRANSFORM은 CREATE TRANSFORM의 반대입니다. 이 명령은 주어진 데이터 유형의 특정 그룹과 연관된 변환 함수를 정의해제합니다. 이러한 그룹과 연관된 함수는 아직 존재하며 여전히 명시적으로 호출할 수 있으나, 더 이상 변환 등록 정보를 가지지 않으므로 호스트 언어 환경과의 값 교환을 위해 내재적으로 호출되지 않습니다.

변환 그룹은 사용자 정의 유형 *type-name*에 대해 정의된 그룹의 변환 함수 중 하나에 종속되며 SQL이 아닌 다른 언어로 작성된 사용자 정의 함수(UDF)가 있는 경우 삭제되지 않습니다(SQLSTATE 42893). 그러한 함수는 유형 *type-name*에 대해 정의된 참조된 변환 그룹과 연관되는 변환 함수에 종속됩니다. 명명된 변환 그룹과 연관된 변환 함수에 종속되는 패키지는 작동 불능으로 표시됩니다.

TRANSFORMS *group-name* **FOR** *type-name*

사용자 정의 데이터 유형 *type-name*에 대해 지정된 변환 그룹이 삭제될 것임을 나타냅니다. 이러한 그룹에서 참조되는 변환 함수는 삭제되지 않습니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용됩니다. 정적 SQL문에서, QUALIFIER 사전 처리 컴파일/바인드 옵션은 내재적으로 규정되지 않은 오브젝트 이름의 규정자를 지정합니다. *type-name*은 카탈로그에 기술된 사용자 정의 유형을 식별해야 하고(SQLSTATE 42704), *group-name*은 *type-name*에 대한 기존의 변환 그룹을 식별해야 합니다.

TRIGGER *trigger-name*

삭제될 트리거를 식별합니다. *trigger-name*은 카탈로그에 기술되어 있는 트리거를 식별해야 합니다(SQLSTATE 42704). 지정된 트리거가 삭제됩니다.

트리거를 삭제하면 일부 패키지가 유효하지 않은 것으로 표시됩니다. 동일한 규칙을 따르는 트리거 작성에 대해서는 879 페이지의 『CREATE TRIGGER』의 "주"절을 참조하십시오.

TYPE *type-name*

삭제할 사용자 정의 유형을 식별합니다. 동적 SQL문에서, CURRENT SCHEMA 특수 레지스터는 규정화되지 않은 오브젝트 이름의 규정자로 사용

됩니다. 정적 SQL문에서, QUALIFIER 사전컴파일/바인드 옵션은 내재적으로 규정화되지 않은 오브젝트 이름의 규정자를 지정합니다. 구조화 유형의 경우, 관련 참조 유형도 삭제됩니다. *type-name*은 카탈로그에 기술된 사용자 정의 유형을 식별해야 합니다. DISTINCT가 지정되면, *type-name*이 카탈로그에서 설명된 구별 유형을 식별해야 합니다. 다음 사항 중 하나라도 참인 경우 유형은 삭제되지 않습니다(SQLSTATE 42893).

- 이 유형이 테이블 또는 뷰 컬럼 유형으로 사용된 경우.
- 유형에 부속 유형이 있는 경우.
- 이 유형이 유형화 테이블 또는 유형화 뷰의 데이터 유형으로 사용된 구조화 유형인 경우.
- 유형이 또다른 구조화 유형의 속성인 경우.
- 유형이 *type-name*의 인스턴스를 포함할 수 있는 테이블의 컬럼이 있는 경우. 이는 *type-name*이 컬럼의 유형이거나 컬럼의 관련 유형 계층에서 사용되는 경우 발생할 수 있습니다. T 유형의 경우, T는 그 유형이 직접 또는 간접적으로 *type-name*을 사용하는 테이블의 컬럼이 있는 경우 삭제될 수 없습니다.
- 유형이 테이블 또는 뷰의 참조 유형 컬럼의 목표 유형이거나 또다른 구조화 유형의 참조 유형 속성인 경우.
- 유형이나 해당 유형에 대한 참조는 삭제할 수 없는 함수 또는 메소드의 매개변수 유형이나 리턴 값 유형입니다.
- 유형이나 해당 유형에 대한 참조는 SQL 함수 또는 메소드의 본문에 사용되지만, 매개변수 유형이나 리턴 값 유형은 아닙니다.
- 유형이 점검 제한조건, 트리거, 뷰 정의 또는 확장 색인에 사용된 경우.

해당 유형을 사용하는 함수: 사용자 정의 유형을 삭제할 수 있는 경우, 삭제 중인 유형 또는 삭제 중인 유형에 대한 참조의 매개변수나 리턴 값을 가지는 모든 함수 F(고유 이름 SF)에 대해 다음의 DROP FUNCTION문이 효과적으로 실행됩니다.

```
DROP SPECIFIC FUNCTION SF
```

이 명령문 또한 종속 함수를 삭제하기 위해 연쇄되는 것이 가능합니다. 이들 모든 함수가 사용자 정의 유형의 종속성으로 인해 삭제될 목록에도 있는 경우, 사용자 정의 유형 삭제가 성공하게 됩니다.(그렇지 않은 경우 SQLSTATE 42893로 실패합니다.)

해당 유형을 사용하는 메소드: 사용자 정의 유형을 삭제할 수 있는 경우, 삭제 중인 유형 또는 삭제 중인 유형에 대한 참조의 매개변수나 리턴 값을 가지는 유형이 T1인 모든 메소드 M(고유 이름 SM)에 대해 다음의 명령문이 효과적으로 실행됩니다.

```
DROP SPECIFIC METHOD SM
ALTER TYPE T1 DROP SPECIFIC METHOD SM
```

이러한 메소드에 종속되는 오브젝트가 있으면 DROP TYPE이 실패할 수 있습니다.

TYPE MAPPING *type-mapping-name*

삭제할 사용자 정의 데이터 유형 맵핑을 식별합니다. *type-mapping-name*은 카탈로그에 기술되어 있는 데이터 유형 맵핑을 식별해야 합니다(SQLSTATE 42704). 데이터 유형 맵핑이 데이터베이스에서 삭제됩니다.

어떠한 추가 오브젝트도 삭제되지 않습니다.

USER MAPPING FOR *authorization-name* | **USER SERVER** *server-name*

삭제될 사용자 맵핑을 식별합니다. 이 맵핑은 연합 데이터베이스를 액세스하기 위해 사용되는 권한 부여 이름을 데이터 소스를 액세스하기 위해 사용된 권한 부여 이름과 연관시킵니다. 이들 두 권한 부여 이름의 첫번째는 *authorization-name*으로 식별되거나 특수 레지스터 USER에 의해 참조됩니다. *server-name*은 액세스하기 위해 두 번째 권한 부여 이름이 사용되는 데이터 소스를 식별합니다.

*authorization-name*은 카탈로그에 나열되어야 합니다(SQLSTATE 42704). *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다 (SQLSTATE 42704). 사용자 맵핑이 삭제됩니다.

어떠한 추가 오브젝트도 삭제되지 않습니다.

VIEW *view-name*

삭제될 뷰를 식별합니다. *view-name*은 카탈로그에 기술되어 있는 뷰

(SQLSTATE 42704)를 식별해야 합니다. 입력된 뷰의 부속 뷰는 상위 뷰에 종속됩니다. 모든 부속 뷰를 삭제해야 상위 뷰를 삭제할 수 있습니다 (SQLSTATE 42893).

지정된 뷰는 삭제됩니다. 그 뷰에 직접 또는 간접으로 종속적인 뷰 또는 트리거의 정의는 작동되지 않는 것으로 표시됩니다. 작동 불능으로 표시된 뷰에 종속적인 요약 테이블이 삭제됩니다. 실행 불능으로 표시되거나 삭제되는 뷰에 종속적인 패키지는 유효하지 않게 됩니다. 여기에는 계층에서 부속 뷰 위에 있는 상위 뷰에 종속된 패키지가 포함됩니다. 삭제된 뷰가 참조 영역으로서 정의된 참조 컬럼의 영역이 해제됩니다.

VIEW HIERARCHY *root-view-name*

삭제될 입력된 뷰 계층을 식별합니다. *root-view-name*은 입력된 뷰 계층에서 루트 뷰인 입력된 뷰를 식별해야 합니다(SQLSTATE 428DR). *root-view-name*에 의해 식별된 입력된 뷰와 모든 서브뷰가 데이터베이스에서 삭제됩니다.

삭제된 뷰에 직접 또는 간접으로 종속적인 뷰 또는 트리거의 정의는 작동 불능으로 표시됩니다. 삭제되거나 작동 불능으로 표시되는 뷰나 트리거에 종속적인 패키지는 무효화될 것입니다. 삭제된 뷰나 작동 불능으로 표시된 뷰에 대한 참조 영역으로서 정의된 참조 컬럼의 영역이 해제됩니다.

WRAPPER *wrapper-name*

삭제될 랩퍼를 식별합니다. *wrapper-name*은 카탈로그에 기술된 랩퍼를 식별해야 합니다(SQLSTATE 42704). 랩퍼가 삭제됩니다.

랩퍼에 종속되는 모든 서버 정의, 사용자 정의 함수(UDF) 맵핑 및 사용자 정의 데이터 유형 맵핑이 삭제됩니다. 삭제된 서버 정의에 종속되는 모든 사용자 정의 함수(UDF) 맵핑, 별명, 사용자 정의 데이터 유형 맵핑 및 사용자 맵핑 역시 삭제됩니다. 삭제된 별명에 종속된 모든 색인 스펙이 삭제되고, 이들 별명에 종속된 모든 뷰가 작동 불능으로 표시됩니다. 삭제된 오브젝트에 종속된 모든 패키지 및 작동 불능 뷰가 무효화됩니다.

규칙

종속성: 1004 페이지의 표27은 종속성을 나타냅니다.[96] 그 오브젝트는 각자에 대해 종속성을 갖습니다. 네 가지의 종속함수가 있습니다.

R 제한 의미. 기반 오브젝트는 그것에 의존하는 오브젝트가 존재하는 한 삭제될 수 없습니다.

C 연쇄 의미. 기반 오브젝트를 삭제하면 그것에 의존하는 오브젝트(종속 오브젝트) 역시 삭제됩니다. 그러나, 종속 오브젝트가 일부 다른 오브젝트 상에서 제한 종속성으로 인해 삭제될 수 없는 경우, 기반 오브젝트 삭제는 실패하게 됩니다.

X 비작동 의미. 기반 오브젝트를 삭제하면, 이 오브젝트에 종속된 오브젝트가 사용할 수 없게 됩니다. 사용자가 명시적인 특정 조치를 취할 때까지 실행 불능 상태입니다.

A 자동 무효화 및 재유효화 의미. 기반 오브젝트를 삭제하면, 이 오브젝트에 종속된 오브젝트가 유효하지 않게 됩니다. 데이터베이스 관리 프로그램은 유효하지 않은 오브젝트를 다시 유효하게 하려는 시도를 합니다.

일부 DROP문 매개변수 및 오브젝트는 공백 행이나 컬럼의 결과를 낳을 수도 있기 때문에 1004 페이지의 표27에 나타나지 않습니다.

- EVENT MONITOR, PACKAGE, PROCEDURE, SCHEMA, TYPE MAPPING 및 USER MAPPING DROP문에는 오브젝트 종속성이 없습니다.
- 별명(Alias), 버퍼 풀, 파티션 키, 특권 및 프로시듀어 오브젝트 유형에는 DROP 문 종속성이 없습니다.
- A DROP SERVER, DROP FUNCTION MAPPING 또는 DROP TYPE MAPPING문에는 다음 조건에서 처리될 수 없는 작업 단위(UOW)가 부여됩니다.
  - 명령문이 단일 데이터 소스를 참조하며, UOW가 이미 데이터 소스 내의 테이블이나 뷰에 대한 별명을 참조하는 SELECT문을 포함합니다(SQLSTATE 55006).

96. 모든 종속성이 카탈로그에 명시적으로 기록되지는 않습니다. 예를 들어, 패키지가 어느 제한조건에 종속하는지에 대한 레코드가 없습니다.

– 명령문이 데이터 소스의 카테고리를 참조하며(예를 들어, 특정 유형 및 버전의 모든 데이터 소스), UOW가 이미 이들 데이터 소스 중 하나 안에 있는 테이블이나 뷰에 대한 별명을 참조하는 SELECT문을 포함합니다(SQLSTATE 55006).

표 27. 종속성

오브젝트 유형 → / 명령문 ↓	CONSTRAINT	FUNCMAPPING	INDEXTENSION	INDEX	METHOD	NICKNAME	PACKAGE	SECURITY LABEL	TABLESPACE	TRIGGER	TYPE MAPPING	USER MAPPING	VIEW
ALTER NICKNAME	-	-	-	-	-	-	-	A	-	-	-	-	-
ALTER SERVER	-	-	-	-	-	-	-	A	-	-	-	-	-
ALTER TABLE DROP CONSTRAINT	C	-	-	-	-	-	-	A[1]	-	-	-	-	-
ALTER TABLE DROP PARTITIONING KEY	-	-	-	-	-	-	R[20]	A[1]	-	-	-	-	-
ALTER TYPE ADD ATTRIBUTE	-	-	R	-	-	-	-	A[23]	-	R[24]	-	-	R[14]
ALTER TYPE DROP ATTRIBUTE	-	-	R	-	-	-	-	A[23]	-	R[24]	-	-	R[14]
ALTER TYPE ADD METHOD	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER TYPE DROP METHOD	-	-	-	-	-	-	-	-	-	-	-	-	-

표 27. 종속성 (계속)

오브젝트 유형 → / 명령문 ↓	CONSTRAINT	FUNCMAPPING	INDEX	INDEXEXTENSION	NICKNAME	NODEGROUP	TABLE	TABLESPACE	TRANSFORM	TRIGGER	TYPEMAPPING	USERMAPPING	VIEW
DROP ALIAS	-	R	-	-	-	-	-	-	A^{3}	-	R^{3}	-	X^{3}	-	-	-	X^{3}
DROP BUFFERPOOL	-	-	-	-	-	-	-	-	-	-	-	R	-	-	-	-	-
DROP FUNCTION	R	R^{7}	R	-	R	R^{7}	-	-	X	-	R	-	R	-	-	-	R
DROP FUNCTION MAPPING	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-
DROP INDEX	R	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	R^{17}
DROP INDEX EXTENSION	-	R	-	R	-	-	-	-	-	-	-	-	-	-	-	-	-
DROP METHOD	R	R^{7}	R	-	R	R	-	-	X	-	R	-	R	-	-	-	R
DROP NICKNAME	-	R	-	C	-	-	-	-	A	-	-	-	-	-	-	-	X^{16}
DROP NODEGROUP	-	-	-	-	-	-	-	-	-	-	-	C	-	-	-	-	-
DROP SERVER	-	C^{21}	C^{19}	-	-	-	C	-	A	-	-	-	-	-	C^{19}	C	-
DROP TABLE	C	R	-	C	-	-	-	-	A^{9}	-	RC^{11}	-	X^{16}	-	-	-	X^{16}
DROP TABLE HIERARCHY	C	R	-	C	-	-	-	-	A^{9}	-	RC^{11}	-	X^{16}	-	-	-	X^{16}
DROP TABLESPACE	-	-	-	C^{6}	-	-	-	-	-	-	CR^{6}	-	-	-	-	-	-
DROP TRANSFORM	-	R	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-

표 27. 종속성 (계속)

오브젝트 유형 → / 명령문 ↓	CONSTRAINT	FUNCMAPPING	INDEXTENSION	METHOD	NICKNAME	NODEGROUP	PACKAGE	SERVER	TABLESPACE	TRIGGER	TYPE	TYPEMAPPING	USERMAPPING	VIEW
DROP TRIGGER	-	-	-	-	-	-	A[1]	-	-	-	-	-	-	-
DROP TYPE	R[13]	R[5]	-	-	R	-	A[12]	-	R[18]	-	R[13]	R[4]	-	-	R[14]
DROP VIEW	-	R	-	-	-	-	A[2]	-	-	-	X[16]	-	-	-	X[15]
DROP VIEW HIERARCHY	-	R	-	-	-	-	A[2]	-	-	-	X[16]	-	-	-	X[16]
DROP WRAPPER	-	-	C	-	-	-	-	C	-	-	-	-	C	-	-
REVOKE 특권[10]	-	CR[25]	-	-	-	-	A[1]	-	CX[8]	-	X	-	-	-	X[8]

1 이 종속성은 이러한 제한조건들, 트리거 또는 파티션 키를 가진 테이블의 종속성으로 인해 내재됩니다.

2 패키지에 뷰에서 활동하는 INSERT, UPDATE, DELETE문이 있는 경우, 그 패키지는 그 뷰의 기초가 되는 기본 테이블에서 삽입, 갱신 또는 삭제를 사용할 수 있습니다. UPDATE의 경우, 이것으로 수정된 기초가 되는 기본 테이블의 각 컬럼에서 갱신을 사용할 수 있습니다.

패키지에 입력된 뷰에 대해 작동하는 명령문이 있는 경우, 동일한 뷰 계층에 있는 뷰를 작성하거나 삭제하면 패키지가 무효가 됩니다.

3 패키지, 요약 테이블, 뷰 또는 트리거가 별명을 사용하는 경우, 이는 별명과 그 별명이 참조하는 오브젝트 모두에 종속됩니다. 별명이 연쇄상에 있는 경우, 연쇄에 있는 각 별명에 대해 종속성이 작성됩니다.

별명 자체는 아무 것에도 종속적이지 않습니다. 존재하지 않는 오브젝트 상에 별명을 정의할 수 있습니다.

4 사용자 정의 유형 T는 다음과 같은 경우 다른 사용자 정의 유형 B에 종속될 수 있습니다.

- B를 속성의 데이터 유형으로 명명
- REF(B) 속성을 가짐
- 상위 유형으로 B를 가짐

5 데이터 유형을 삭제하면 연쇄적으로 해당 데이터 유형을 이에 대해 정의된 매개변수, 결과 유형 및 메소드로서 사용하는 함수와 메소드가 삭제됩니다. 이러한 함수와 메소드의 삭제는 서로 종속적이라는 사실만으로 예방되지는 않습니다. 그러나 그 내용에서 해당 데이터 유형을 사용하는 함수나 메소드의 경우에는 제한사항이 적용됩니다.

6 테이블 공간이나 테이블 공간의 목록을 삭제하면 주어진 테이블공간이나 목록 내에 포함된 모든 테이블이 삭제됩니다. 그러나, 한 테이블이 테이블 공간에 걸쳐 있고(서로 다른 테이블 공간에 있는 색인이나 긴 컬럼), 그러한 테이블 공간이 삭제 중인 목록에 있지 않는 경우에는, 테이블이 존재하는 한 테이블 공간이 삭제될 수 없습니다.

7 종속 함수가 SOURCE절에 있는 기본 함수를 명명하는 경우, 함수가 다른 특정 함수에 종속될 수 있습니다. 함수나 메소드는 종속되는 루틴이 SQL로 작성되고 그 내용에 기본 루틴을 사용하는 경우 다른 특정 함수나 메소드에 종속적일 수 있습니다. 구조화 유형 매개변수나 리턴 유형을 갖는 외부 메소드나 외부 함수는 하나 이상의 변환 함수에 종속적이 됩니다.

8 SELECT 특권이 없어지면 삭제될 요약 테이블이나 뷰가 작동 불능 상태로 됩니다. 실행 불능 상태가 된 뷰가 입력된 뷰 계층에 포함될 경우, 그에 속한 모든 부속 뷰도 실행 불능 상태가 됩니다.

9 패키지에 테이블 T 에서 활동하는 INSERT, UPDATE, DELETE문이 있는 경우, 그 패키지는 T에서 삽입, 갱신 또는 삭제를 사용할 수 있습니다. UPDATE의 경우, 이것으로 수정된 기초가 되는 기본 테이블 T의 각 컬럼에서 갱신을 사용할 수 있습니다.

패키지에 입력된 테이블에 대해 작동하는 명령문이 있는 경우, 동일한 테이블 계층에 있는 테이블을 작성하거나 삭제하면 패키지가 무효가 됩니다.

10 컬럼의 특권은 개별적으로 취소될 수 없으므로, 컬럼 레벨의 종속성은 존재하지 않습니다.

모든 서브테이블이나 서브뷰상의 SELECT 특권에 대한 종속성이 있습니다.패키지, 트리거 또는 뷰가 FROM절에서 OUTER(*Z*)의 사용을 포함하면, *Z*의 모든 서브테이블이나 서브뷰상의 SELECT 특권에 대한 종속성이 있습니다. 마찬가지로, 패키지, 트리거 또는 뷰가 DEREF(*Y*)의 사용을 포함하면(여기서 *Y*는 테이블이나 뷰 *Z*와의 참조 유형), *Z*의 모든 서브테이블이나 서브뷰상의 SELECT 특권에 대한 종속성이 있습니다.

11 요약 테이블은 기본 테이블 또는 테이블 정의의 fullselect에 지정된 테이블에 종속됩니다.

연쇄 의미론이 종속되는 요약 테이블에 적용됩니다.

서브테이블은 최대 루트 테이블에 이르기까지 그의 수퍼테이블들에 종속적입니다. 모든 서브테이블을 삭제할 때까지 상위 테이블은 삭제할 수 없습니다.

12 패키지는 TYPE 술어나 부속 유형 처리 표현식(TREAT *expression* AS *data-type*)을 사용한 결과로서 구조화 유형에 종속적일 수 있습니다. 패키지는 TYPE 술어 오른쪽 또는 TREAT 표현식의 오른쪽 부분에 지정된 각 구조화 유형의 부속 유형에 종속적입니다. 패키지가 종속된 부속 유형을 교체하는 구조화 유형을 삭제하거나 작성하면 무효가 됩니다.

13 제한조건이나 트리거에서 유형이 사용되는 경우, 점검 제한조건 또는 트리거는 유형에 종속적입니다. 점검 제한조건 또는 트리거 내부의 TYPE 술어에 사용되는 구조화 유형의 부속 유형에 종속성이 없습니다.

14 뷰 정의(입력된 뷰의 유형 포함)에서 유형이 사용되는 경우 뷰는 유형에 종속적입니다. 뷰 정의 내부의 TYPE 술어에 사용되는 구조화 유형의 부속 유형에 종속성이 없습니다.

15 부속 뷰는 그 상위 뷰(루트 뷰까지)에 종속됩니다. 모든 부속 뷰를 삭제할 때까지 상위 뷰는 삭제할 수 없습니다. 추가 뷰 종속성에 대해서는 [16]에서 참조하십시오.

16 트리거나 뷰는 참조 해제 연산 또는 DEREF 함수의 목표 테이블 또는 목표 뷰에도 종속적입니다. OUTER(*Z*)를 포함하는 FROM절이 있는 트리거 또는 뷰는, 트리거나 뷰가 작성될 때 있었던 *Z*의 모든 서브테이블 또는 부속 뷰에 종속적입니다.

17 오브젝트 식별자 컬럼의 고유성을 확인하기 위해, 입력된 뷰는 고유 색인 존재에 종속될 수 있습니다.

18 테이블은 사용자 정의 데이터 유형이 다음과 같기 때문에 이 유형(구별 또는 구조화)에 종속될 수 있습니다.

- 컬럼의 유형으로 사용된 경우.
- 테이블의 유형으로 사용된 경우.
- 테이블 유형의 속성으로 사용된 경우.
- 테이블의 컬럼 유형이나 테이블의 유형 속성인 참조 유형의 목표 유형으로 사용된 경우.
- 직접 또는 간접적으로 테이블의 컬럼인 유형에 의해 사용되는 경우.

19 서버를 삭제하면 그 명명된 서버에 작성된 함수 맵핑 및 유형 맵핑도 연쇄적으로 삭제됩니다.

20 다중 파티션 노드그룹에 있는 테이블에 파티션 키가 정의되면, 파티션 키가 필요합니다.

21 종속 OLE DB 테이블 함수가 ″R″ 종속 오브젝트를 가지는 경우(DROP FUNCTION 참조), 서버를 삭제할 수 없습니다.

22 SQL 함수나 메소드는 그 내용에서 참조하는 오브젝트에 종속될 수 있습니다.

23 *type-name* T인 유형 TA의 속성 A를 삭제했다면 다음 DROP문이 효과적으로 실행됩니다.

```
Mutator method: DROP METHOD A (TA) FOR T
Observer method: DROP METHOD A () FOR T
ALTER TYPE T
   DROP METHOD A(TA)
   DROP METHOD A()
```

[24] 테이블은 다음 경우에 사용자 정의 구조화 데이터 유형의 속성에 종속될 수 있습니다.

1. 해당 테이블이 *type-name* 또는 그의 부속 유형에 기초하는 유형화 테이블인 경우.
2. 테이블에 직접 또는 간접적으로 *type-name*을 참조하는 유형의 기존 컬럼이 있는 경우.

[25] SQL 함수의 내용에 사용되는 테이블이나 뷰에 대한 SELECT 특권을 취소하는 경우 정의된 해당 함수에 더 이상 SELECT WITH GRANT OPTION 특권이 없다면 해당 함수가 삭제됩니다. 이러한 함수는 뷰나 트리거에 사용될 경우 삭제할 수 없으며 REVOKE는 결과로 제한됩니다. 그렇지 않으면 REVOKE가 연쇄적으로 이러한 함수를 삭제합니다.

주

- 사용자 정의 함수를 사용중인 동안 삭제할 수 있습니다. 또한, 커서는 사용자 정의 함수를 참조하는 명령문상에서 열릴 수 있으며, 이 커서가 열려 있는 동안 함수는 그 커서의 페치를 실패하지 않고도 삭제할 수 있습니다.
- 사용자 정의 함수에 종속되는 패키지를 실행하는 경우, 패키지가 현재의 작업 단위(UOW)를 완료할 때까지는 또다른 ID로 함수를 삭제할 수 없습니다. 그 시점에서, 함수는 삭제되고, 패키지는 실행 불능 상태가 됩니다. 다음에 이 패키지에 대한 요청을 하면, 패키지가 명시적으로 리바인드되어야 함을 나타내는 오류가 발생합니다.
- 함수 내용의 제거(함수의 삭제와는 전혀 다름)가 이 함수 내용을 필요로 하는 응용프로그램의 실행중에 발생할 수 있습니다. 이로 인해 명령문이 실패할 수도 있는데, 이는 명령문을 대신해서 데이터베이스 관리 프로그램이 함수 내용을 저장영역에 적재할 필요가 있는 지에 따라 다릅니다.
- DATALINK 컬럼을 통해 현재 링크된 파일을 포함하는 삭제된 테이블인 경우, 파일 링크가 취소된 후 데이터 링크 컬럼 정의에 따라 복원되거나 삭제됩니다.
- 데이터베이스에 구성된 임의의 DB2 Data Links Manager가 사용불가능할 때 DROP TABLE 또는 DROP TABLESPACE를 통해 DATALINK 컬럼을 포함하는 테이블이 삭제되면, 조작은 실패합니다(SQLSTATE 57050).

- 변환이 내재적으로 필요한 경우에는 명시적으로 지정된 UDF에 대해 기록된 종속성에 추가로 다음 종속성도 기록됩니다.
  1. 함수나 메소드의 구조화 유형 매개변수 또는 결과에 변환이 필요한 경우, 필요한 TO SQL 또는 FROM SQL 변환 함수에 대해 해당 함수나 메소드의 종속성이 기록됩니다.
  2. 패키지에 포함된 SQL문이 변환 함수를 필요로 하는 경우에는 지정된 TO SQL이나 FROM SQL 변환 함수에 대해 해당 패키지의 종속성이 기록됩니다.

  위에서는 변환의 내재적 호출로 인해 종속성이 기록되는 환경만을 기술하였으므로, 함수, 메소드 또는 패키지가 아닌 다른 오브젝트는 내재적으로 호출된 변환 함수에 대한 종속성을 가질 수 없습니다. 다시 말해서, 변환 함수에 대한 명시적 호출(예를 들어, 뷰 및 트리거에서)의 결과 이러한 유형의 다른 오브젝트가 변환 함수에 종속됩니다. 결과적으로, DROP TRANSFORM문은 삭제 중인 변환 함수에 대한 오브젝트의 이러한 "명시적" 유형 종속성으로 인해 실패할 수도 있습니다(SQLSTATE 42893).
- 종속성 카탈로그는 변환 함수로서 함수에 종속되는지와 명시적 함수 호출에 의해 함수에 종속되는지를 구별하지 않으므로, 변환 함수에 대한 명시적 호출은 작성되지 않는다고 가정합니다. 이 경우 함수에 대한 변환 등록 정보를 삭제할 수 없거나 패키지가 작동 불능으로 표시되는데, 이는 단순히 SQL 표현식에 명시적 호출을 포함하기 때문입니다.

예

예 1: TDEPT 테이블을 삭제합니다.

```
DROP TABLE TDEPT
```

예 2: VDEPT를 삭제합니다.

```
DROP VIEW VDEPT
```

예 3: 권한 부여 ID, HEDGES가 별명을 삭제하려 합니다.

```
DROP ALIAS A1
```

별명 HEDGES.A1은 카탈로그에서 제거됩니다.

예 4: 경계가 별명 삭제를 시도하지만, T1을 별명으로서 지정합니다. 여기서 T1은(별명이 아닌) 기존 테이블 이름입니다.

```
DROP ALIAS T1
```

이 명령문은 실패합니다(SQLSTATE 42809).

예 5:

BUSINESS_OPS 노드 그룹을 삭제하십시오. 노드 그룹을 삭제하려면, 노드 그룹의 두 테이블 공간(ACCOUNTING 및 PLANS)이 우선 삭제되어야 합니다.

```
DROP TABLESPACE ACCOUNTING
DROP TABLESPACE PLANS
DROP NODEGROUP BUSINESS_OPS
```

예 6: Pellow는 삭제할 함수 인스턴스를 식별하기 위한 시그니처를 사용하여 PELLOW 스키마에 작성한 CENTRE 함수를 삭제하려 합니다.

```
DROP FUNCTION CENTRE (INT,FLOAT)
```

예 7: McBride는 삭제할 함수 인스턴스를 식별하기 위한 특정 이름을 사용하여 PELLOW 스키마에 작성한 FOCUS92 함수를 삭제하려 합니다.

```
DROP SPECIFIC FUNCTION  PELLOW.FOCUS92
```

예 8: CHEM 스키마의 함수 ATOMIC_WEIGHT를 삭제합니다. 이 스키마에는 이 이름을 갖는 함수가 하나 뿐입니다.

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT
```

예 9: 사원들이 특정 조건하에서 급여에 대한 상여금을 받도록 한 트리거 SALARY_BONUS를 삭제합니다.

```
DROP TRIGGER SALARY_BONUS
```

예 10: shoesize라는 구별 데이터 유형을 현재 사용하지 않는 경우, 이를 삭제합니다.

```
DROP DISTINCT TYPE SHOESIZE
```

예 11: SMITHPAY 이벤트 모니터를 삭제합니다.

```
DROP EVENT MONITOR SMITHPAY
```

예 12: RESTRICT을 사용하는 CREATE SCHEMA에 따라 예 2에서 스키마 삭제. PART 테이블을 우선 삭제해야 합니다.

```
DROP TABLE PART
DROP SCHEMA INVENTRY RESTRICT
```

예 13: Macdonald는 삭제할 프로시듀어 인스턴스를 식별하기 위한 특정 이름을 사용하여 EIGLER 스키마에 작성한 DESTROY 프로시듀어를 삭제하려 합니다.

```
DROP SPECIFIC PROCEDURE  EIGLER.DESTROY
```

예 14: BIOLOGY 스키마에서 프로시듀어 OSMOSIS를 삭제합니다. 이 스키마에는 이 이름을 갖는 프로시듀어가 하나 뿐입니다.

```
DROP PROCEDURE BIOLOGY.OSMOSIS
```

예 15: 사용자 SHAWN이 하나의 권한 부여 ID를 연합 데이터베이스를 액세스하는 데 사용하고, 다른 것을 ORACLE1이라고 하는 Oracle 데이터 소스를 액세스하는 데 사용했습니다. 두 권한 부여 사이에 맵핑이 작성되었으나, SHAWN은 더 이상 데이터 소스를 액세스할 필요가 없습니다. 맵핑을 삭제하십시오.

```
DROP USER MAPPING FOR SHAWN SERVER ORACLE1
```

예 16: 별명이 참조하는 데이터 소스 테이블의 색인이 삭제되었습니다. 최적화 알고리즘이 이 색인에 대해 알게 하도록 작성된 색인 스펙을 삭제하십시오.

```
DROP INDEX INDEXSPEC
```

예 17: MYSTRUCT1 변환 그룹을 삭제하십시오.

```
DROP TRANSFORM MYSTRUCT1 FOR POLYGON
```

예 18: PERSONNEL 스키마에서 EMP 데이터 유형의 메소드 BONUS를 삭제하십시오.

```
DROP METHOD BONUS (SALARY DECIMAL(10,2)) FOR PERSONNEL.EMP
```

END DECLARE SECTION

END DECLARE SECTION문은 호스트 변수 선언 섹션의 끝을 표시합니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 실행되지 않습니다. REXX에 지정되어서는 안 됩니다.

권한 부여

필요 사항 없음.

구문

설명

END DECLARE SECTION문은 선언문이 호스트 언어 규칙에 따라 나타날 수 있는 모든 응용프로그램에서 코드화될 수 있습니다. 이는 호스트 변수 선언 섹션의 끝을 표시합니다. 호스트 변수 절은 BEGIN DECLARE SECTION문으로 시작합니다(574 페이지의 『BEGIN DECLARE SECTION』 참조).

BEGIN DECLARE SECTION문 및 END DECLARE SECTION문은 쌍을 이루어야 하나, 중첩될 수는 없습니다.

호스트 변수 선언문은 SQL INCLUDE문을 사용하여 지정할 수 있습니다. 그렇지 않은 경우, 호스트 변수 선언 섹션에는 호스트 변수 선언문 이외의 다른 명령문이 포함되어서는 안 됩니다.

SQL문에서 참조되는 호스트 변수는 REXX가 아닌 다른 모든 호스트 언어로 된 호스트 변수 선언 섹션에서 선언되어야 합니다.[97] 추가로 각 변수의 선언은 해당 변수에 대한 첫번째 참조에 앞서 나타나야 합니다.

97. LOB 위치 지정자 및 파일 참조 변수에 대해 호스트 변수를 REXX로 선언하는 방법에 대해서는 574 페이지의 『규칙』을 참조하십시오.

선언 섹션 외부에서 선언된 변수는 선언 섹션 내에서 선언된 변수 이름과 같아서는 안 됩니다.

예

END DECLARE SECTION문을 사용하는 예에 대해 574 페이지의 『BEGIN DECLARE SECTION』에서 참조하십시오.

EXECUTE

EXECUTE문은 준비된 SQL문을 실행합니다.

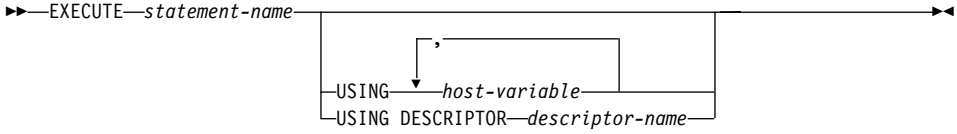
호출

이 명령문은 응용프로그램에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

명령문(DDL, GRANT 및 REVOKE문) 실행시 권한 부여 점검이 수행될 경우, 명령문의 권한 부여 ID가 부여하는 특권은 PREPARE문이 지정한 SQL문을 실행하는 데 필요한 명령문을 포함해야 합니다. 명령문(DML) 준비시, 권한 부여 점검이 수행되는 명령문의 경우, 이 명령문을 사용하는 데 필요한 권한이 없습니다.

구문



설명

statement-name

실행될 준비 명령문을 식별합니다. *statement-name*은 이전에 준비된 명령문을 식별해야 하며, 준비된 명령문은 SELECT문이면 안 됩니다.

USING

준비된 명령문에 있는 매개변수 표시문자(의문부호)에 대한 값이 대치될 호스트 변수 목록이 표시됩니다.(매개변수 표시문자에 대해 1087 페이지의 『PREPARE』에서 참조하십시오.) 준비된 명령문에 매개변수 표시문자가 있는 경우, USING을 사용해야 합니다.

host-variable, ...

호스트 변수 선언 규칙에 따라 프로그램에 선언되어 있는 호스트 변수를 식별합니다. 변수의 수는 준비된 명령문의 매개변수 표시문자 수와 같아야

합니다. *n*번째 변수는 준비된 명령문의 *n*번째 매개변수에 일치합니다. 위치 지정자 변수 및 파일 참조 변수는, 해당되는 경우, 매개변수 표시문자 값의 소스로서 제공될 수 있습니다.

DESCRIPTOR *descriptor-name*

유효한 호스트 변수 설명이 들어 있는 입력 SQLDA를 식별합니다.

EXECUTE문이 처리되기 전에 다음 필드를 입력 SQLDA에 설정해야 합니다.

- SQLDA에 제공되는 SQLVAR 어커런스 수를 나타내기 위한 SQLN.
- SQLDA에 대해 할당되는 저장영역의 바이트 수를 나타내기 위한 SQLDABC.
- 명령문 처리시 SQLDA에서 사용되는 변수의 수를 나타내기 위한 SQLDA.
- 변수의 속성을 나타내기 위한 SQLVAR 어커런스 수.

SQLDA는 모든 SQLVAR 어커런스가 들어가기에 충분한 저장영역이 있어야 합니다. 따라서 SQLDABC의 값은 16 + SQLN*(N)보다 크거나 같아야 합니다. 여기서 N은 SQLVAR 발생의 길이입니다.

LOB 입력 데이터를 조정해야 하는 경우, 모든 매개변수 표시문자에 대해 두 개의 SQLVAR 항목이 있어야 합니다.

SQLD는 0 이상이거나 SQLN 이하의 값으로 설정되어야 합니다. 1267 페이지의 『부록C. SQL 설명자 영역(SQLDA)』에서 자세한 정보를 참조하십시오.

주

- 준비된 명령문이 실행되기 전에, 각 매개변수 표시문자는 해당 호스트 변수 값으로 대체됩니다. 입력된 매개변수 표시문자의 경우, 목표 변수 속성은 CAST 스펙으로 지정된 속성입니다. 미입력 매개변수 표시문자의 경우, 목표 변수 속성은 매개변수 표시문자의 문맥에 따라 결정됩니다. 매개변수 표시문자에 영향을 주는 규칙에 관해서는 1088 페이지의 『규칙』에서 참조하십시오.

매개변수 표시문자 P에 해당하는 호스트 변수로 V를 지정하도록 하십시오. V 값은 컬럼에 값을 지정하는 규칙에 따라 P에 대한 목표 변수에 할당됩니다. 따라서,

- V는 목표와 호환성이 있어야 합니다.
- V가 문자열인 경우, 그 길이는 목표 속성 길이보다 작아야 합니다.
- V가 숫자인 경우, 정수 부분의 절대값은 목표의 정수 부분의 최대 절대값 길이보다 작아야 합니다.
- V 속성이 목표 속성과 같지 않은 경우, 그 값은 목표 속성에 일치하도록 변환됩니다.

준비된 명령문이 실행될 때, P 대신 사용된 값은 P에 대한 목표 변수의 값입니다. 예를 들어, V가 CHAR(6)이고 목표가 CHAR(8)인 경우, P 대신에 사용된 값은 두 개의 공백이 채워진 V의 값입니다.

- ***동적 SQL문 캐쉬***

  정적 SQL문이 첫번째로 참조되거나 동적 SQL문이 첫번째로 준비될 경우, 동적 및 정적 SQL문을 실행하는 데 필요한 정보는 데이터베이스 패키지 캐쉬에 보관되어 있습니다. 이 정보는 데이터베이스 종료할 때, 다른 명령문에서 캐쉬 공간을 필요로 할때, 정보가 유효하지 않을 때까지 패키지 캐쉬에 남아 있습니다.

  SQL문이 실행되거나 준비되면, 응용프로그램 발행 요청과 관련된 패키지 정보는 시스템 카탈로그에서 패키지 캐쉬로 로드됩니다. 개개의 SQL문에 대해 실제로 실행되는 섹션은 캐쉬에 위치됩니다. 명령문이 첫번째로 참조되거나 동적 SQL 섹션이 작성된 후에 직접 캐쉬에 위치되면, 시스템 카탈로그로에서 정적 SQL 섹션이 읽혀지고 패키지에 위치됩니다. 동적 SQL 섹션은 PREPARE 또는 EXECUTE IMMEDIATE문과 같은 명시적 명령문으로 작성할 수 있습니다. 일단 작성되면 원래의 섹션이 공간 관리 이유로 삭제되거나 환경 변경으로 인해 유효하지 않게 될 경우, 동적 SQL문에 대해 섹션은 시스템이 수행한 내재적으로 준비된 명령문으로 재작성할 수 있습니다.

  각 SQL문은 데이터베이스 레벨에서 캐쉬되며, 응용프로그램들 사이에서 공유할 수 있습니다. 정적 SQL문은 동일한 응용프로그램들 사이에서 공유할 수 있습니다; 동적 SQL문은 아주 동일한 명령문 텍스트와 동일한 컴파일 환경으로 응용프로그램들 사이에서 공유할 수 있습니다. 응용프로그램이 발행한 각 SQL문

의 텍스트는 내재적 준비가 필요한 이벤트에 사용하기 위해 응용프로그램에 지역적으로 캐쉬됩니다. 응용프로그램의 PREPARE문은 각각 하나의 명령문을 캐쉬할 수 있습니다. 응용프로그램의 모든 EXECUTE IMMEDIATE문은 캐쉬 공간을 공유하며, 모든 EXECUTE IMMEDIATE문에 대해 한 번에 하나의 캐쉬문만이 존재합니다. 동일한 PREPARE 또는 임의의 EXECUTE IMMEDIATE 문이 항상 다른 SQL문으로 여러 번 발행되면, 마지막 명령문만이 재사용을 위해 캐쉬에 보관됩니다. 캐쉬의 선택적 사용으로 응용프로그램이 한 번 시작되고 나면 필요에 따라 EXCUTE 또는 OPEN문과 서로 다른 번호의 PREPARE문이 발행될 것입니다.

캐쉬된 동적 SQL문을 사용함에 있어, 일단 명령문이 작성되면, 명령문을 다시 준비할 필요없이 여러 작업 단위(UOW)에서 재사용할 수 있습니다. 시스템은 환경이 변경되는 경우, 필요에 따라 명령문을 다시 컴파일합니다.

다음과 같은 환경 또는 데이터 오브젝트의 변경으로 인해, 내재적으로 준비될 동적 명령문이 다음 PREPARE, EXECUTE, EXECUTE IMMEDIATE 또는 OPEN 요청에서 캐쉬될 것입니다.

- ALTER NICKNAME
- ALTER SERVER
- ALTER TABLE
- ALTER TABLESPACE
- ALTER TYPE
- CREATE FUNCTION
- CREATE FUNCTION MAPPING
- CREATE INDEX
- CREATE TABLE
- CREATE TEMPORARY TABLESPACE
- CREATE TRIGGER
- CREATE TYPE
- DROP (모든 오브젝트)
- 테이블 또는 색인의 RUNSTATS
- 뷰가 실행 불능이 되도록 하는 모든 조치

- UPDATE (시스템 카탈로그 테이블내 통계)
- SET CURRENT DEGREE
- SET PATH
- SET QUERY OPTIMIZATION
- SET SCHEMA
- SET SERVER OPTION

다음 목록은 캐쉬된 동적 SQL문으로부터 예상할 수 있는 사항을 요약한 것입니다.

- *PREPARE 요청 :* 이 후의 동일한 명령문 준비로 인해, 섹션이 여전히 유효한 경우 명령문을 컴파일하는 데 비용이 발생하지 않게 됩니다. 현재의 캐쉬 섹션에 대해 산정한 비용과 기본 행수가 리턴됩니다. 이 값은 동일한 SQL문에 대해 이전에 PREPARE로부터 리턴된 값과 다릅니다.

  COMMIT 또는 ROLLBACK문 이후에 PREPARE문을 발행할 필요가 없을 것입니다.
- *EXECUTE 요청:* EXECUTE문이 이후에 원래의 PREPARE 이후에 유효하지 않게 되면, 명령문을 내재적으로 준비하는 데 비용이 발생합니다. 내재적으로 준비된 섹션은 현재 환경에서는 사용되지만 원래의 PREPARE문 환경에서는 사용되지 않습니다.
- *EXECUTE IMMEDIATE 요청:* 동일한 명령에 대해 이 후의 EXECUTE IMMEDIATE문은 섹션이 여전히 유효한 경우, 명령문을 컴파일하는 데 비용이 발생하지 않습니다.
- *OPEN 요청* 동적으로 정의된 커서에 대한 OPEN 요청은 원래의 PREPARE 이후 유효하지 않게 되었으면 명령문을 내재적으로 준비하는 데 비용이 발생합니다. 내재적으로 준비된 섹션은 현재 환경에서는 사용되지만 원래의 PREPARE문 환경에서는 사용되지 않습니다.
- *FETCH 요청:* 어떠한 작동의 변경도 없을 것입니다.
- *ROLLBACK:* 작업 단위가 구간 복원 조작에 영향을 끼치는 동안 준비 또는 내재적으로 준비된 동적 SQL문만 무효화됩니다.
- *COMMIT:* 동적 SQL문은 무효화되지 않지만 확보된 잠금은 해제됩니다. WITH HOLD 커서로 정의되지 않은 커서는 닫히고, 그 잠금은 해제됩니다.

열려 있는 WITH HOLD 커서는 패키지와 섹션 잠금을 유지하여 확약 프로세스 중 또는 후에 활성 섹션을 보호합니다.

내재적 준비중 오류가 발생하면, 내재적 준비를 일으킨 요청에 대해 오류를 리턴합니다.

예

예 1: 이 C의 예에서, 매개변수 표시문자를 가진 INSERT문이 준비되고 실행됩니다. h1 - h4는 TDEPT 형식에 해당하는 호스트 변수입니다.

```
strcpy (s,"INSERT INTO TDEPT VALUES(?,?,?,?)");
EXEC SQL PREPARE DEPT_INSERT FROM :s;
   .
   .
(Check for successful execution and put values into :h1, :h2, :h3, :h4)
   .
   .
EXEC SQL EXECUTE DEPT_INSERT USING :h1, :h2,
:h3, :h4;
```

예 2: 이 EXECUTE문은 SQLDA를 사용합니다.

```
EXECUTE S3 USING DESCRIPTOR :sqlda3
```

EXECUTE IMMEDIATE

EXECUTE IMMEDIATE문:

- 명령문의 문자열로부터 SQL문의 실행 가능 형식을 준비합니다.
- SQL문을 실행합니다.

EXECUTE IMMEDIATE는 PREPARE 및 EXECUTE문의 기본 함수를 결합합니다. 호스트 변수나 매개변수 표시문자 모두가 들어 있지 않은 SQL문을 준비하고 실행하는 데 사용될 수 있습니다.

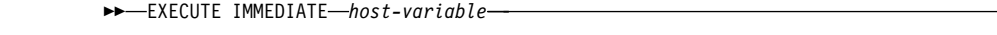
호출

이 명령문은 응용프로그램에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

권한 부여 규칙은 EXECUTE IMMEDIATE에 의해 지정된 SQL문에 대해 정의된 규칙입니다.

구문



설명

host-variable

호스트 변수가 지정되어 문자열 변수 선언 규칙에 따라 프로그램 내에 기술되는 호스트 변수를 식별해야 합니다. 최대 명령문 길이 65 535보다 작은 문자열 변수이어야 합니다. CLOB(65535)는 최대 크기 명령문을 포함할 수 있으나 VARCHAR은 포함할 수 없습니다.

식별된 호스트 변수 값은 명령문 문자열이라고 합니다.

명령문 문자열은 다음 SQL문 중 하나여야 합니다.

- ALTER
- COMMENT ON

- COMMIT
- CREATE
- DELETE
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- GRANT
- INSERT
- LOCK TABLE
- REFRESH TABLE
- RELEASE SAVEPOINT
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- ROLLBACK
- SAVEPOINT
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET CURRENT TRANSFORM GROUP
- SET EVENT MONITOR STATE
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION
- UPDATE

명령문 문자열에는 매개변수 표시문자나 호스트 변수에 대한 참조가 포함되지 않아야 하며, EXEC SQL로 시작해서는 안 됩니다. 트리거 SQL문을 구분하기 위한 CREATE TRIGGER문(세미콜론을 포함할 수 있음)이나 SQL 프로시듀어 본문에서 SQL문을 구분하기 위한 CREATE PROCEDURE문을 제외한 명령문 종료자는 포함될 수 없습니다.

EXECUTE IMMEDIATE문이 실행될 때, 지정된 문자열이 분석되고 오류 검사가 실시됩니다. SQL문이 유효하지 않은 경우, 실행되지 않고 실행을 막는 오류 조건이 SQLCA에 보고됩니다. SQL문이 유효하나 실행 중에 오류가 발생하는 경우, 그 오류 조건은 SQLCA에 보고됩니다.

주

- 명령문 캐쉬는 EXECUTE IMMEDIATE문의 실행 방식에 영향을 미칩니다. 1018 페이지의 『동적 SQL문 캐쉬』에서 정보를 참조하십시오.

예

C 프로그램 명령문을 사용하여 SQL문을 호스트 변수 qstring(char[80])로 이동시킨 후, 호스트 변수 qstring에 있는 SQL문을 실행합니다.

```
if ( strcmp(accounts,"BIG") == 0 )
  strcpy (qstring,"INSERT INTO WORK_TABLE SELECT *
    FROM EMP_ACT WHERE ACTNO < 100");
 else
  strcpy (qstring,"INSERT INTO WORK_TABLE SELECT *
    FROM EMP_ACT WHERE ACTNO >= 100");
  .
  .
  .
EXEC SQL  EXECUTE IMMEDIATE :qstring;
```

EXPLAIN

EXPLAIN문은 제공된 설명 가능한 명령문에 선택된 액세스 플랜에 관한 정보를 보관하고, 이 정보를 Explain 테이블에 둡니다. (Explain 테이블 및 테이블 정의에 대해 1455 페이지의 『부록K. Explain 테이블 및 정의』에서 정보를 참조하십시오.)

*설명 가능 명령문*은 DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES 또는 VALUES INTO SQL문입니다.

호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

설명될 명령문은 실행되지 않습니다.

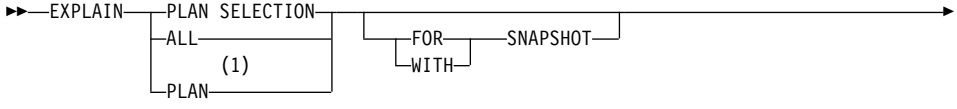
권한 부여

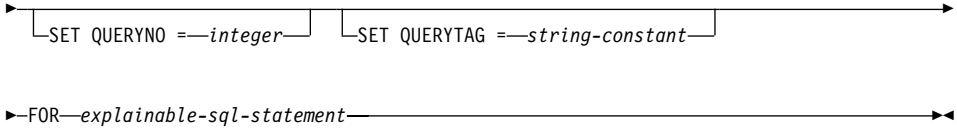
권한 부여 규칙은 EXPLAIN문에 지정된 SQL문에 대해 정의된 규칙입니다. 예를 들어, *explainable-sql-statement*으로서 DELETE문이 사용된 경우(다음에 나오는 명령문 구문 참조), DELETE문 설명시 DELETE문에 대한 권한 부여 규칙이 적용됩니다.

정적 EXPLAIN문에 대한 권한 부여 규칙은 *explainable-sql-statement*으로 통과되는 명령문의 정적 버전에 적용되는 규칙입니다. 동적으로 준비된 EXPLAIN문은 *explainable-sql-statement* 매개변수에 제공된 명령문의 동적 준비를 위한 권한 부여 규칙을 사용합니다.

현재의 권한 부여 ID는 'Explain' 테이블에 삽입 특권을 가져야 합니다.

구문





주:

1 PLAN 옵션은 MVS EXPLAIN문에 대한 기존 DB2의 구문 허용 한도에 대해서만 지원됩니다. PLAN 테이블은 없습니다. PLAN을 지정하는 것은 PLAN SELECTION을 지정하는 것과 같습니다.

설명

PLAN SELECTION

SQL 컴파일의 플랜 선택 단계 정보가 Explain 테이블로 삽입된다는 것을 나타냅니다.

ALL

ALL을 지정하는 것은 PLAN SELECTION을 지정하는 것과 같습니다.

PLAN

PLAN 옵션은 다른 시스템에서 기존의 데이터베이스 응용프로그램에 대해 구문 허용 한도를 제공합니다. PLAN을 지정하는 것은 PLAN SELECTION을 지정하는 것과 같습니다.

FOR SNAPSHOT

이 절은 Explain 스냅샷만이 취해져 EXPLAIN_STATEMENT 테이블의 SNAPSHOT 컬럼에 위치한다는 것을 나타냅니다. EXPLAIN_INSTANCE 및 EXPLAIN_STATEMENT 테이블에 존재하는 것 이외의 다른 Explain 정보가 보관되지 않습니다.

Explain 스냅샷 정보는 Visual Explain에 사용하기 위한 것입니다.

WITH SNAPSHOT

이 절은 일반적인 Explain 정보 외에도 Explain 스냅샷이 취해진다는 것을 나타냅니다.

EXPLAIN문의 기본 작동은 Explain 스냅샷이 아닌 일반적인 Explain 정보만 수집하는 것입니다.

Explain 스냅샷 정보는 Visual Explain에 사용하기 위한 것입니다.

기본값(FOR SNAPSHOT과 WITH SNAPSHOT이 모두 지정되지 않았음)
Explain 테이블에 Explain 정보를 둡니다. Visual Explain에 사용할 스냅샷이 취해지지 않습니다.

SET QUERYNO = *integer*
EXPLAIN_STATEMENT 테이블의 QUERYNO 컬럼을 통해 *integer*를 *explainable-sql-statement*과 연결시킵니다. 제공되는 정수 값은 양수여야 합니다.

이 절이 동적 EXPLAIN문에 지정되어 있지 않으면, (1)의 기본값이 지정됩니다. 정적 EXPLAIN문의 경우, 지정되지 않은 기본값은 사전 처리 컴파일러가 지정한 명령문 번호입니다.

SET QUERYTAG = *string-constant*
EXPLAIN_STATEMENT 테이블의 QUERYTAG를 통해서 *string-constant*을 *explainable-sql-statement*에 연결시킵니다. *string-constant*은 최고 20바이트의 문자열일 수 있습니다. 제공된 값의 길이가 20바이트 보다 작은 경우, 값은 필수 길이까지 오른쪽에 공백이 채워집니다.

이 절이 EXPLAIN문에 지정되어 있지 않으면, 기본값으로 공백이 사용됩니다.

FOR *explainable-sql-statement*
설명될 SQL문을 지정합니다. 이 명령문은 유효한 DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES 또는 VALUES INTO SQL문이 될 수 있습니다. EXPLAIN문이 프로그램에 포함되어 있으면, *explainable-sql-statement*에는 호스트 변수(프로그램에 정의되어 있어야 함)에 대한 참조가 포함될 수 있습니다. 마찬가지로 EXPLAIN이 동적으로 준비되고 있는 경우, *explainable-sql-statement*에는 매개변수 표시문자가 포함될 수 있습니다.

*explainable-sql-statement*은 EXPLAIN문과 독립적으로 준비되고 실행될 수 있는 유효한 SQL문이어야 합니다. 이것은 명령문 이름이나 호스트 변수여서는 안 됩니다. CLP를 통해 정의된 커서를 참조하는 SQL문은 이 명령문에 사용하기에 적합하지 않습니다.

응용프로그램 내의 동적 SQL을 Explain하려면, 전체 EXPLAIN문이 동적으로 준비되어야 합니다.

주

다음 테이블은 스냅샷 키워드와 Explain 정보와의 상호 관계를 보여줍니다.

Keyword Specified	Capture Explain Information?	Take Snapshot for Visual Explain?
none	Yes	No
FOR SNAPSHOT	No	Yes
WITH SNAPSHOT	Yes	Yes

FOR SNAPSHOT 및 WITH SNAPSHOT절이 모두 지정되지 않은 경우 Explain 스냅샷을 취할 수 없습니다.

Explain 테이블은 EXPLAIN을 호출하기 전에 사용자가 작성해야 합니다. (Explain 테이블 및 테이블 정의에 대해 1455 페이지의 『부록K. Explain 테이블 및 정의』에서 정보를 참조하십시오.) 이 명령문에 의해 생성된 정보는 명령문이 컴파일될 때 지정된 스키마에 있는 이와 같은 Explain 테이블에 저장됩니다.

제공된 *explainable-sql-statement*의 컴파일 중 오류가 발생하면, Explain 테이블에 정보간 저장되지 않습니다.

*explainable-sql-statement*에 대해 생성된 액세스 플랜은 보관되지 않으므로, 나중에 호출할 수 없습니다. EXPLAIN문 자체가 컴파일될 때 *explainable-sql-statement*에 대한 Explain 정보가 삽입됩니다.

정적 EXPLAIN SQL문의 경우, 정보는 바인드시와 명시적 리바인드 중에 Explain 테이블에 삽입됩니다(*Command Reference*에서 REBIND 참조). 사전 처리 컴파일되는 동안 정적 EXPLAIN문은 수정된 적용업무 원시 파일에 주석처리됩니다. 바인드될 때 EXPLAIN문은 SYSCAT.STATEMENTS 카탈로그에 저장됩니다. 패키지가 실행될 때 EXPLAIN문은 실행되지 않습니다. 응용프로그램에 있는 모든 명령문에 대한 섹션 번호는 연속적이어야 하며, EXPLAIN문이 포함되어야 함에 유의하십시오. 정적 EXPLAIN문 사용의 대체 방법은 EXPLAIN 및 EXPLSNAP BIND/PREP 옵션의 조합을 사용하는 것입니다. 정적 EXPLAIN문을 사용하여 여

러 개 중 하나의 정적 SQL에 Explain 테이블이 이식되게 할 수 있습니다. 적절한 EXPLAIN문 구문을 사용하여 목표 명령문에 접두어를 붙인 다음 Explain BIND/PREP 옵션을 사용하지 않고 응용프로그램을 바인드하십시오. 또한, EXPLAIN문은 실제 Explain 호출시 QUERYNO 또는 QUERYTAG 필드를 설정하는 것이 유리한 경우에 사용될 수도 있습니다.

증분식 바인드 EXPLAIN SQL문의 경우, Explain 테이블은 EXPLAIN문이 컴파일을 위해 제출될 때 상주하게 됩니다. 패키지가 수행될 때 EXPLAIN문은 처리를 수행하지 않습니다(명령문이 성공될지라도). Explain 테이블의 데이터를 처리하는 경우, 데이터 처리 중에 사용되는 Explain 테이블 규정자와 권한 부여 ID는 패키지 소유자의 규정자 및 권한 부여 ID입니다. 또한 EXPLAIN문은 실제 Explain 호출시 QUERYNO 또는 QUERYTAG 필드를 설정하는 것이 유리한 경우에 사용될 수도 있습니다.

동적 EXPLAIN문의 경우, Explain 테이블은 EXPLAIN문이 컴파일을 위해 제출될 때 상주하게 됩니다. Explain문은 PREPARE문으로 준비할 수 있으나, 실행되는 경우 처리되지 않습니다(명령문이 성공적일지라도). 동적 EXPLAIN문을 실행하는 대신 CURRENT EXPLAIN MODE와 CURRENT EXPLAIN SNAPSHOT 특수 레지스터를 사용하여 동적 SQL문을 설명할 수 있습니다. EXPLAIN문은 실제 Explain 호출시 QUERYNO 또는 QUERYTAG 필드를 설정하는 것이 유리한 경우에 사용되어야 합니다.

예

예 1: 간단한 SELECT문을 설명하고 QUERYNO = 13이라는 태그를 표시합니다.

```
EXPLAIN PLAN SET QUERYNO = 13 FOR SELECT C1 FROM T1;
```

이 명령문은 성공적입니다.

예 2:

간단한 SELECT문을 설명하고 QUERYTAG = 'TEST13' 표시를 합니다.

```
EXPLAIN PLAN SELECTION SET QUERYTAG = 'TEST13'
  FOR SELECT C1 FROM T1;
```

이 명령문은 성공적입니다.

예 3: 간단한 SELECT문을 설명하고 QUERYNO = 13과 QUERYTAG = 'TEST13'으로 태그를 붙입니다.

```
EXPLAIN PLAN SELECTION SET QUERYNO = 13 SET QUERYTAG = 'TEST13'
  FOR SELECT C1 FROM T1;
```

이 명령문은 성공적입니다.

예 4: Explain 테이블이 존재하지 않을 때 Explain 정보를 확보하려 합니다.

```
EXPLAIN ALL FOR SELECT C1 FROM T1;
```

이 명령문은 Explain 테이블이 정의되어 있지 않으므로 실패하게 됩니다 (SQLSTATE 42704).

FETCH

FETCH문은 결과 테이블의 다음 행에 커서를 위치시키고, 호스트 변수에 그 행의 값을 지정합니다.

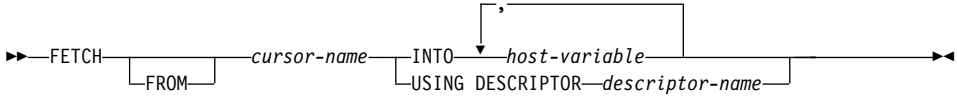
호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

커서를 사용하는 데 필요한 권한에 대해 952 페이지의『DECLARE CURSOR』에서 참조하십시오.

구문



설명

cursor-name

페치(fetch)에 사용되는 커서를 식별합니다. *cursor-name*은 952 페이지의『DECLARE CURSOR』에 설명된 대로 선언된 커서를 식별해야 합니다. DECLARE CURSOR문은 원시 프로그램에서 FETCH문 앞에 위치해야 합니다. FETCH문이 실행될 때, 커서는 열린 상태에 있어야 합니다.

커서가 현재 결과 테이블의 마지막 행 이후에 있는 경우,

- SQLCODE는 +100으로 설정되고, SQLSTATE는 '02000'으로 설정됩니다.
- 커서는 마지막 행 후에 위치합니다.
- 호스트 변수에 값이 지정되지 않습니다.

현재 커서가 행 앞에 위치한 경우, 그 행에 커서가 재위치되고 값은 INTO 또는 USING에 의해 지정된 대로 호스트 변수에 지정됩니다.

커서가 현재 마지막 행이 아닌 행에 위치한 경우, 다음 행에 커서가 재위치되고, 그 행의 값은 INTO 또는 USING에 의해 지정된 대로 호스트 변수에 지정됩니다.

INTO *host-variable*, **...**

호스트 변수 선언 규칙에 따라 기술되어 있는 여러 개의 호스트 변수를 식별합니다. 결과 행의 첫번째 값은 목록의 첫번째 호스트 변수에 지정되고, 두 번째 값은 두 번째 호스트 변수에, 이와 같은 식으로 지정됩니다. 선택 목록에 있는 LOB 값의 경우, 목표는 일반 호스트 변수(충분히 큰 경우), 위치 지정자 변수 또는 파일 참조 변수가 될 수 있습니다.

USING DESCRIPTOR *descriptor-name*

0개 이상의 호스트 변수의 유효한 설명이 들어 있는 SQLDA를 식별합니다.

FETCH문이 처리되기 전에, 사용자는 다음 필드를 SQLDA에 설정해야 합니다.

- SQLDA에 제공된 SQLVAR 어컨런스 수를 표시하는 SQLN
- SQLDA에 할당된 저장영역의 바이트 수를 표시하는 SQLDABC
- 명령문 처리시 SQLDA에 사용된 변수의 수를 표시하는 SQLD
- 변수의 속성을 나타내기 위한 SQLVAR 어커런스 수.

SQLDA는 모든 SQLVAR 어커런스가 들어가기에 충분한 저장영역이 있어야 합니다. 따라서 SQLDABC의 값은 16 + SQLN*(N)보다 크거나 같아야 합니다. 여기서 N은 SQLVAR 발생의 길이입니다.

LOB 또는 구조화 유형 결과 컬럼을 조절해야 하는 경우, 모든 선택 목록 항목(또는 결과 테이블의 컬럼)에 대해 두 개의 SQLVAR 항목이 있어야 합니다. SQLDOUBLED, LOB 및 구조화된 유형 컬럼에 대해서는 1274 페이지의 『SQLDA에서 DESCRIBE의 영향』에서 참조하십시오.

SQLD는 0 이상이거나 SQLN 이하의 값으로 설정되어야 합니다. 1267 페이지의 『부록C. SQL 설명자 영역(SQLDA)』에서 자세한 정보를 참조하십시오.

INTO절에 의해 식별되거나 SQLDA에 설명되어 있는 *n*번째 변수는 커서의 결과 테이블 *n*번째 컬럼에 해당합니다. 각 변수의 데이터 유형은 해당 컬럼과 일치해야 합니다.

『제3장 언어 요소』에 설명되어 있는 규칙에 따라 변수가 각각 지정됩니다. 변수의 수가 그 행의 값의 수보다 적은 경우, SQLDA의 SQLWARN3 필드는 'W'로 설정됩니다. 결과 컬럼의 수보다 많은 변수가 있는 경우, 경고가 없음에 유의하십시오. 지정 오류가 발생하는 경우, 변수에 값이 지정되지 않고 변수에 더 이상의 값이 지정되지 않습니다. 이미 변수에 지정된 값은 지정된 상태로 남아 있습니다.

주

- 열린 커서에는 다음과 같이 세가지의 위치를 갖을 수 있습니다.
  - 행 앞에
  - 행에
  - 마지막 행 이후
- 커서가 행에 있는 경우, 그 행을 그 커서의 현재 행이라고 합니다. UPDATE 또는 DELETE문에 참조되는 커서는 행에 위치해야 합니다. 커서는 FETCH문의 결과로서 행에만 있을 수 있습니다.
- FETCH문을 통해 위치 지정자를 보유하는 것이 필요없는 상황에서 LOB 위치 지정자를 검색할 때, 위치 지정자 자원이 제한되어 있으므로 다음 FETCH문을 발행하기 전에 FREE LOCATOR문을 발행하는 것이 좋습니다.
- 커서를 예측할 수 없는 상태로 만드는 오류가 발생할 수 있습니다.
- FETCH에 대해 경고가 리턴되지 않을 수도 있습니다. 리턴된 경고가 이전에 페치된 행에 적용될 수도 있습니다. 이는 SYSTEM TEMPORARY 테이블이나 푸쉬다운 연산자 사용과 같은 최적화의 결과로서 발생합니다(*관리 안내서* 참조).
- 명령문 캐쉬는 EXECUTE IMMEDIATE문의 실행 방식에 영향을 미칩니다. 1017 페이지의 『주』에서 자세한 정보를 참조하십시오.
- DB2 CLI는 추가의 패치 능력을 지원합니다. 예를 들어, 커서의 결과 테이블이 읽기 전용일 경우, SQLFetchScroll() 함수를 사용하여 커서를 결과 테이블 내의 아무 지점에 위치시킬 수 있습니다.

예

예 1: 이 예에서, FETCH문은 SELECT문의 결과를 프로그램 변수 dnum, dname 및 mnum으로 페치합니다. 페치할 행이 더 이상 없으면, 찾기 불가능 조건으로 돌아갑니다.

```
EXEC SQL  DECLARE C1 CURSOR FOR
   SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
    WHERE ADMRDEPT = 'A00';
EXEC SQL  OPEN C1;
 while (SQLCODE==0) {
   EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
 }
EXEC SQL  CLOSE C1;
```

예 2: 이 FETCH문은 SQLDA를 사용합니다.

```
FETCH CURS USING DESCRIPTOR :sqlda3
```

FLUSH EVENT MONITOR

FLUSH EVENT MONITOR문은 이벤트 모니터 *event-monitor-name*과 연관된 모든 사용중 모니터 유형에 대한 현재 데이터베이스 모니터 값을 이벤트 모니터 I/O 목표에 씁니다. 그러므로, 언제든 부분적 이벤트 레코드가 낮은 레코드 생성 빈도를 가진 이벤트 모니터(데이터베이스 이벤트 모니터와 같은)에 사용 가능합니다. 그러한 레코드는 *부분적 레코드* 식별자로 이벤트 모니터 로그에 기록됩니다.

이벤트 모니터가 지워질 때, 사용중인 내부 버퍼들이 이벤트 모니터 출력 오브젝트에 쓰여집니다.

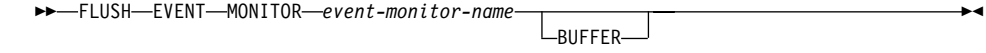
호출

이 명령문은 응용프로그램에 넣거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

권한 부여 ID에서 갖고 있는 특권에는 SYSADM 또는 DBADM 권한을 포함해야 합니다(SQLSTATE 42502).

구문



설명

event-monitor-name

이벤트 모니터의 이름. 이 이름은 한 부분의 이름입니다. 이는 SQL 식별자입니다.

BUFFER

이벤트 모니터 버퍼가 쓰여질 것임을 나타냅니다. BUFFER를 지정하면 부분적 레코드가 생성되지 않습니다. 이벤트 모니터 버퍼에 이미 존재하는 데이터만이 쓰여집니다.

주

- 이벤트 모니터를 비우면 이벤트 모니터 값이 재설정되지 않을 것입니다. 이는 비우기가 수행되지 않은 경우, 생성된 이벤트 모니터 레코드가 정상 모니터 이벤트가 트리거될 때에도 여전히 생성될 것임을 나타냅니다.

FREE LOCATOR

FREE LOCATOR문은 위치 지정자 변수와 그 값 사이의 연관을 제거합니다.

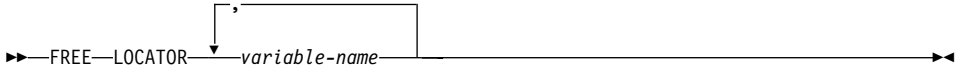
호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

필요 사항 없음.

구문



설명

LOCATOR *variable-name,* **...**

위치 지정자 변수 선언 규칙에 따라 선언되어야 하는 여러 위치 지정자 변수를 식별합니다.

위치 지정자 변수에는 현재 그것에 지정된 위치 지정자가 있어야 합니다. 즉, 이 작업 단위(UOW) 내에서 (FETCH문 또는 SELECT INTO문을 통해) 위치 지정자가 지정되어야 하며, (FREE LOCATOR문을 사용하여) 지정을 연속적으로 해제하면 안 됩니다. 그렇지 않으면 오류가 발생합니다(SQLSTATE 0F001).

하나 이상의 위치 지정자가 지정되면, 목록의 다른 위치 지정자에서 검출되는 오류에 관계없이 제거될 수 있는 모든 위치 지정자가 제거됩니다.

예

COBOL 프로그램에서, BLOB 위치 지정자 변수 TKN-VIDEO와 TKN-BUF, 그리고 CLOB 위치 지정자 변수 LIFE-STORY-LOCATOR를 제거합니다.

```
EXEC SQL
FREE LOCATOR :TKN-VIDEO, :TKN-BUF, :LIFE-STORY-LOCATOR
END-EXEC.
```

GRANT(데이터베이스 권한)

이러한 GRANT문 형태는(데이터베이스 내의 특정 오브젝트에 적용되는 특권이 아닌) 전체 데이터베이스에 적용되는 권한을 부여합니다.

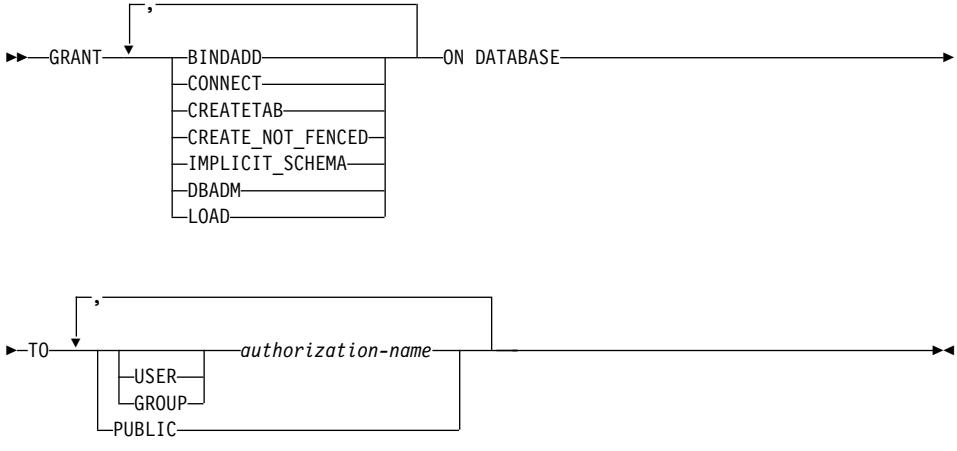
호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

DBADM 권한을 부여하려면, SYSADM 권한이 필요합니다. 다른 권한을 부여하려면, DBADM 또는 SYSADM 권한이 필요합니다.

구문



설명

BINDADD

패키지를 만들 권한을 부여합니다. 패키지 작성자는 자동으로 그 패키지에 대한 CONTROL 특권을 가지며, 나중에 BINDADD 권한이 취소되어도 이 특권을 유지합니다.

CONNECT

데이터베이스에 액세스할 권한을 부여합니다.

CREATETAB

기본 테이블을 작성할 권한을 부여합니다. 기본 테이블 작성자는 자동으로 그 테이블에 대한 CONTROL 특권을 가지게 됩니다. 작성자는 후에 CREATETAB 권한이 취소되어도 이 특권을 유지합니다.

뷰 작성에 필요한 특별한 권한은 없습니다. 뷰 작성에 사용되는 명령문의 권한 부여 ID가 그 뷰의 각 기본 테이블에 CONTROL 또는 SELECT 특권이 있는 경우 언제라도 뷰를 작성할 수 있습니다.

CREATE_NOT_FENCED

데이터베이스 관리 프로그램 프로세스에서 실행되는 레지스터 함수에 권한을 부여합니다. 그렇게 등록된 함수는 역부가작용을 하지 않도록 주의해야 합니다 (자세히 알려면 668의 FENCED 또는 NOT FENCED절 참조).

일단 함수가 안정되지 않은 것으로 등록되면, 나중에 CREATE_NOT_FENCED가 권한 취소되어도 이러한 방식으로 계속 실행됩니다.

IMPLICIT_SCHEMA

스키마를 내재적으로 작성하기 위한 권한을 부여합니다.

DBADM

데이터베이스 관리자 권한을 부여합니다. 데이터베이스 관리자는 데이터베이스에 있는 모든 오브젝트에 대해 모든 특권을 가지며, 이들 특권을 다른 사용자에게 권한 부여할 수도 있습니다.

BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED 및 IMPLICIT_SCHEMA는 DBADM 권한이 부여된 *authorization-name*에 자동으로 제공됩니다.

LOAD

이 데이터베이스에서 로드할 권한을 부여합니다. 이 권한은 사용자에게 이 데이터베이스의 로드 유틸리티를 사용할 수 있는 권한을 제공합니다. SYSADM 및 DBADM도 기본값으로 이러한 권한을 가집니다. 그러나 사용자가 LOAD

권한(SYSADM 또는 DBADM이 아닌)만을 가지는 경우, 사용자는 테이블 레벨 특권을 가져야 합니다. LOAD 특권에 추가로 사용자는 다음을 가져야 합니다.

- INSERT, TERMINATE (이전의 LOAD INSERT를 종료) 또는 RESTART(이전 LOAD INSERT를 재시작) 모드로 LOAD 테이블에 대한 INSERT 특권
- REPLACE, TERMINATE(이전의 LOAD REPLACE를 종료) 또는 RESTART(이전 LOAD INSERT를 재시작) 모드로 LOAD 테이블에 대한 INSERT 및 DELETE 특권
- 그러한 테이블이 LOAD 일부로 사용되는 경우 예외 테이블에 대한 INSERT 특권

TO

권한이 부여되는 대상을 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

모든 사용자에게 권한을 부여합니다. DBADM은 PUBLIC으로 권한 부여될 수 없습니다.

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  - 권한 부여 이름이 운영 시스템에 GROUP으로만 정의되어 있으면, GROUP으로 취급됩니다.

- 권한 부여 이름이 운영 시스템에 USER로만 정의되어 있거나 정의되어 있지 않으면, USER로 취급됩니다.
- 권한 부여 이름이 운영 시스템에서 둘다로 정의되거나 DCE 인증이 사용되면, 오류(SQLSTATE 56092)가 발생합니다.

예

예 1: 사용자 WINKEN, BLINKEN 및 NOD에게 데이터베이스에 연결하기 위한 권한을 제공합니다.

```
GRANT CONNECT ON DATABASE TO USER WINKEN, USER BLINKEN, USER NOD
```

예 2: 데이터베이스에 대한 BINDADD 권한을 그룹 D024에 부여합니다. 시스템에 D024란 사용자와 그룹이 모두 있습니다.

```
GRANT BINDADD ON DATABASE TO GROUP D024
```

GROUP 키워드가 지정되어야 함에 유의하십시오. 그렇지 않은 경우, D024라는 사용자와 그룹이 모두 존재하므로 오류가 발생합니다. D024 그룹 구성원은 데이터베이스의 패키지를 바인드할 수 있도록 허용되어 있으나, (이 사용자 또한 D024 그룹의 구성원이고 이전에 BINDADD 권한을 부여받았거나, D024가 구성원이었던 다른 그룹에 BINDADD 권한이 부여되었던 경우를 제외하고는) D024 사용자는 허용되지 않습니다.

GRANT(색인 특권)

이러한 형태의 GRANT문은 색인에 대한 CONTROL 특권을 권한 부여합니다.

호출

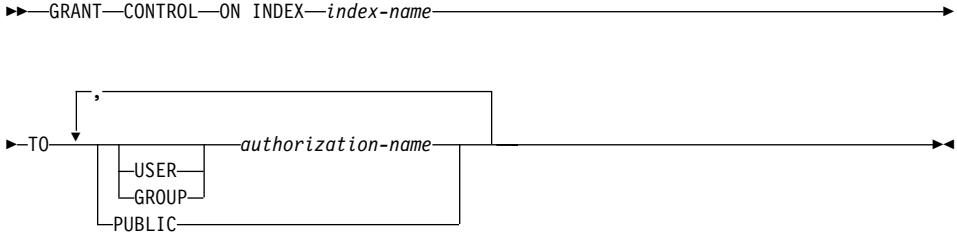
이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- DBADM 권한
- SYSADM 권한

구문



설명

CONTROL

색인을 삭제할 특권을 권한 부여합니다. 이는 색인에 대한 CONTROL 권한으로서, 색인 작성자에게 자동으로 권한 부여됩니다.

ON INDEX *index-name*

CONTROL 특권이 권한 부여될 색인을 식별합니다.

TO

특권이 권한 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

모든 사용자에게 특권을 권한 부여합니다.

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  - 권한 부여 이름이 운영 시스템에 GROUP으로만 정의되어 있으면, GROUP으로 취급됩니다.
  - 권한 부여 이름이 운영 시스템에 USER로만 정의되어 있거나 정의되어 있지 않으면, USER로 취급됩니다.
  - 권한 부여 이름이 운영 시스템에서 둘다로 정의되거나 DCE 인증이 사용되면, 오류(SQLSTATE 56092)가 발생합니다.

예

```
GRANT CONTROL ON INDEX DEPTIDX TO USER USER4
```

GRANT(패키지 특권)

이러한 형태의 GRANT문은 패키지에 대한 특권을 권한 부여합니다.

호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

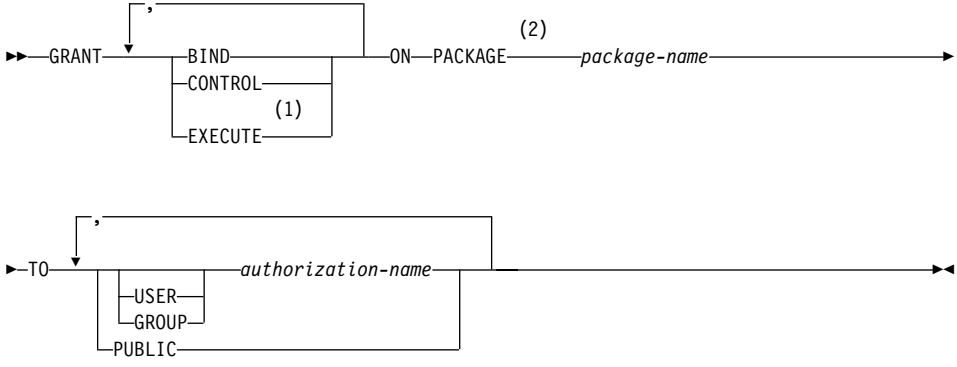
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- 참조된 패키지에 대한 CONTROL 특권
- SYSADM 또는 DBADM 권한

CONTROL 특권을 권한 부여하려면, SYSADM 또는 DBADM 권한이 필요합니다.

구문



주:

1 RUN은 EXECUTE에 대한 동의어로 사용할 수 있습니다.

2 PACKAGE에 대한 동의어로 PROGRAM을 사용할 수 있습니다.

설명

BIND

패키지를 바인드할 특권을 권한 부여합니다. BINDADD 권한을 가진 사람에 의해 패키지가 이미 바인드되었으므로 BIND 특권은 진정한 리바인드 특권입니다.

사용자는 BIND 특권 이외에도 프로그램 내의 정적 DML문에서 참조하는 각 테이블에 대해 필요한 특권을 소유해야 합니다. 정적 DML문에 대한 권한이 바인드될 때 검사되므로 이는 반드시 필요합니다.

CONTROL

패키지 리바인드, 삭제 또는 실행 특권을 권한 부여하고 다른 사용자에게 패키지 특권을 확장시킵니다. 이는 색인에 대한 CONTROL 특권으로서, 색인 작성자에게 자동으로 권한 부여됩니다. 패키지 소유자는 패키지 바인더 또는 바인드/사전 처리 컴파일시 OWNER 옵션으로 지정된 ID입니다.

BIND 및 EXECUTE는 CONTROL 특권이 권한 부여된 *authorization-name*에게 자동으로 권한 부여됩니다.

EXECUTE

패키지를 실행할 특권을 권한 부여합니다.

ON PACKAGE *package-name*

특권이 권한 부여될 패키지 이름을 지정합니다.

TO

특권이 권한 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

모든 사용자에게 특권을 권한 부여합니다.

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  - *authorization-name*이 운영 시스템에 GROUP으로만 정의되어 있으면, GROUP으로 취급됩니다.
  - *authorization-name*이 운영 시스템에 USER로만 정의되어 있거나 정의되어 있지 않으면, USER로 취급됩니다.
  - *authorization-name*이 운영 시스템에서 둘다로 정의되거나 DCE 인증이 사용되면, 오류(SQLSTATE 56092)가 발생합니다.

예

예 1: 패키지 CORPDATA.PKGA에 대한 EXECUTE 특권을 PUBLIC에 권한 부여합니다.

```
GRANT EXECUTE
   ON PACKAGE CORPDATA.PKGA
 TO PUBLIC
```

예 2: 패키지 CORPDATA.PKGA에 대한 GRANT EXECUTE 특권을 EMPLOYEE라는 사용자에게 권한 부여합니다. EMPLOYEE라는 사용자나 그룹은 없습니다.

```
GRANT EXECUTE ON PACKAGE
   CORPDATA.PKGA TO EMPLOYEE
```

또는

```
GRANT EXECUTE ON PACKAGE
   CORPDATA.PKGA TO USER EMPLOYEE
```

GRANT(스키마 특권)

이러한 형태의 GRANT문은 스키마에 대한 특권을 권한 부여합니다.

호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

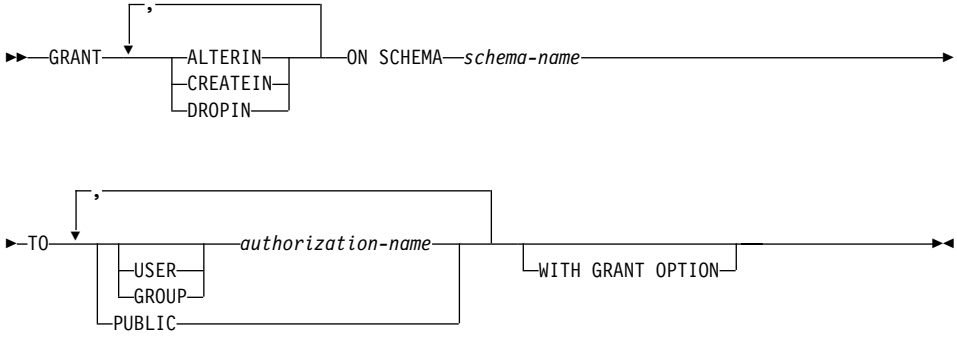
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- *schema-name*의 식별된 각 특권에 대한 WITH GRANT OPTION
- SYSADM 또는 DBADM 권한

스키마 이름 SYSIBM, SYSCAT, SYSFUN 및 SYSSTAT에는 어떤 사용자도 특권을 권한 부여할 수 없습니다.

구문



설명

ALTERIN

스키마에 있는 모든 오브젝트에 주석을 달거나 변경할 수 있는 특권을 권한 부여합니다. 명시적으로 작성된 스키마의 소유자는 자동으로 ALTERIN 특권을 부여받습니다.

CREATEIN

스키마에 오브젝트를 작성할 특권을 권한 부여합니다. 그래도 오브젝트를 작성하는 데 필요한 기타 권한이나 특권(예: CREATETAB)이 있어야 합니다. 명시적으로 작성된 스키마의 소유자는 자동으로 CREATEIN 특권을 부여받습니다. 내재적으로 작성된 스키마는 CREATEIN 특권을 자동으로 PUBLIC에 권한 부여합니다.

DROPIN

스키마의 모든 오브젝트를 삭제하기 위한 특권을 권한 부여합니다. 명시적으로 작성된 스키마의 소유자는 자동으로 DROPIN 특권을 부여받습니다.

ON SCHEMA *schema-name*

특권이 권한 부여될 스키마를 식별합니다.

TO

특권이 권한 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

모든 사용자에게 특권을 권한 부여합니다.

WITH GRANT OPTION

지정된 *권한 부여 이름*이 다른 사람에게 특권을 권한 부여할 수 있도록 합니다.

WITH GRANT OPTION가 생략되면, 지정된 *권한 부여 이름*은 다음의 경우에만 다른 사람에게 특권을 권한 부여할 수 있습니다.

- DBADM 권한을 갖거나
- 다른 소스의 특권을 권한 부여하기 위한 능력을 제공받은 경우

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  - 권한 부여 이름이 운영 시스템에 GROUP으로만 정의되어 있으면, GROUP으로 취급됩니다.
  - 권한 부여 이름이 운영 시스템에 USER로만 정의되어 있거나 정의되어 있지 않으면, USER로 취급됩니다.
  - 권한 부여 이름이 운영 시스템에서 둘다로 정의되거나 DCE 인증이 사용되면, 오류(SQLSTATE 56092)가 발생합니다.
- 일반적으로, GRANT문은 명령문의 권한 부여 ID가 권한 부여하도록 허용된 특권 권한 부여 프로세스를 수행하며, 두 개 이상의 특권이 권한 부여되지 않은 경우 경고(SQLSTATE 01007)를 리턴합니다. 아무 특권도 부여되지 않으면, 오류가 리턴됩니다(SQLSTATE 42501).[98]

예

예 1: 스키마 CORPDATA에 오브젝트를 작성하기 위한 능력을 USER2에 권한 부여합니다.

```
GRANT CREATEIN ON SCHEMA CORPDATA TO USER2
```

예 2: 스키마 CORPDATA에 오브젝트를 작성 및 삭제하기 위한 능력을 사용자 BIGGUY에게 권한 부여합니다.

98. 명령문 처리에 사용된 패키지가 MIA에 대해 SQL92E로 설정된 LANGLEVEL로 사전 처리 컴파일되었으면, 권한 부여자에게 해당 권한의 오브젝트에 대한 특권이 없는한 경고가 리턴됩니다(SQLSTATE 01007).

```
GRANT CREATEIN, DROPIN ON SCHEMA CORPDATA TO BIGGUY
```

GRANT(서버 특권)

이러한 형태의 GRANT문은 통과 모드에서 지정된 데이터 소스를 액세스하고 사용할 특권을 권한 부여합니다.

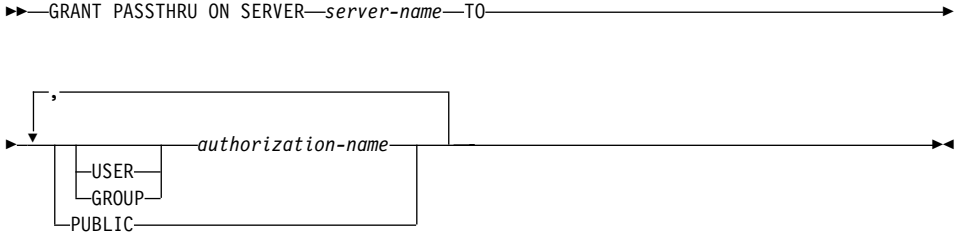
호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID는 SYSADM 또는 DBADM 권한을 가집니다.

구문



설명

server-name

통과 모드에서 사용할 특권이 권한 부여되고 있는 데이터 소스에 이름을 부여합니다. *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다.

TO

특권을 권한 부여한 사용자로 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

모든 사용자에게 *server-name*을 통과하는 특권을 권한 부여합니다.

예

예 1: 데이터 소스 SERVALL을 통과하는 특권이 있는 Give R. Smith 및 J. Jones. 그들의 권한 부여 ID는 RSMITH와 JJONES입니다.

```
GRANT PASSTHRU ON SERVER SERVALL
   TO USER RSMITH,
   USER JJONES
```

예 2: 권한 부여 ID가 D024인 그룹에 데이터 소스 EASTWING을 통과할 특권을 권한 부여. 권한 부여 ID 또한 D024인 사용자가 있습니다.

```
GRANT PASSTHRU ON SERVER EASTWING TO GROUP D024
```

GROUP 키워드가 지정되어야 합니다. 그렇지 않으면, D024가 지정된 그룹의 ID일 뿐만 아니라 사용자의 ID이기 때문에 오류가 발생합니다(SQLSTATE 56092). 그룹 D024의 어떠한 구성원이든 EASTWING에 통과하도록 허용될 것입니다. 그러므로, 사용자 D024가 그룹에 속하면, 이 사용자는 EASTWING로 통과할 수 없습니다.

GRANT(테이블, 뷰 또는 별명 특권)

이러한 형태의 GRANT문은 테이블, 뷰 또는 별명에 대한 특권을 부여합니다.

호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

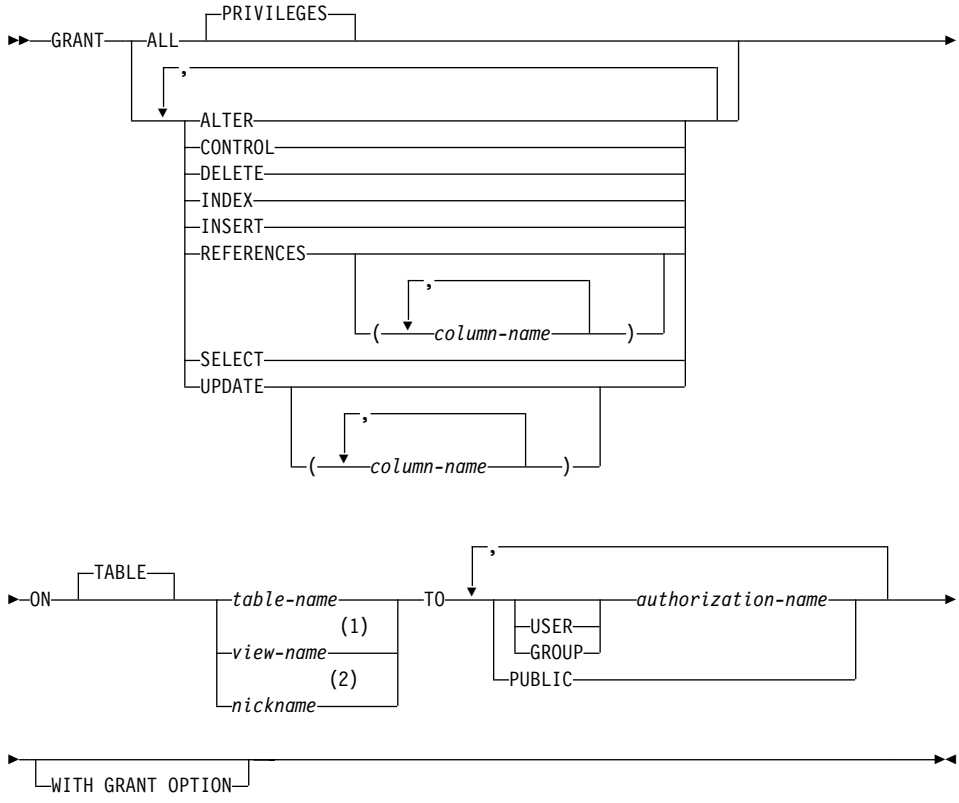
명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- 참조된 테이블, 뷰 또는 별명에 대한 CONTROL 특권
- 식별된 각 특권에 대한 WITH GRANT OPTION. ALL이 지정된 경우, 권한 부여 ID는 식별된 테이블, 뷰 또는 별명에 대해 어느 정도의 부여 가능 특권이 있어야 합니다.
- SYSADM 또는 DBADM 권한

CONTROL 특권을 권한 부여하려면, SYSADM 또는 DBADM 권한이 필요합니다.

카탈로그 테이블 및 뷰에 대한 특권을 권한 부여하려면, SYSADM 또는 DBADM 권한이 필요합니다.

구문



주:

1 ALTER, INDEX 및 REFERENCES 특권은 뷰에 적용되지 않습니다.

2 DELETE, INSERT, SELECT 및 UPDATE 특권은 별명에 적용 가능하지 않습니다.

설명

ALL 또는 **ALL PRIVILEGES**

ON절에 명명된 기본 테이블, 뷰 또는 별명에 CONTROL을 제외한 적절한 모든 특권을 부여합니다.

명령문의 권한 부여 ID가 테이블, 뷰 또는 별명에 대한 CONTROL 특권이나 DBADM 또는 SYSADM 권한을 갖는 경우, 오브젝트에 적용할 수 있는 (CONTROL을 제외한) 모든 특권이 권한 부여됩니다. 그렇지 않으면, 권한 부

여된 특권은 명령문의 권한 부여 ID가 식별된 테이블, 뷰 또는 별명에 대해 갖는 권한 부여 가능한 모든 특권입니다.

ALL이 지정되지 않으면, 특권 목록에 있는 하나 이상의 키워드를 반드시 지정해야 합니다.

ALTER

다음 사항에 특권을 권한 부여합니다.

- 기본 테이블 정의에 컬럼을 추가.
- 기본 테이블에 대한 기본 키나 고유성 제한조건을 작성 또는 삭제합니다. 기본 키나 고유성 제한조건을 작성 또는 삭제하는 데 필요한 권한에 대해 524 페이지의 『ALTER TABLE』에서 더 자세한 정보를 참조하십시오.
- 기본 테이블에 외부 키를 작성 또는 삭제.

  상위 테이블의 각 컬럼에 대한 REFERENCES 특권도 필요합니다.
- 기본 테이블에 점검 제한조건을 작성 또는 삭제.
- 기본 테이블에 트리거를 작성.
- 별명에 대한 컬럼 추가, 재설정 또는 삭제 옵션.
- 별명 컬럼 이름이나 데이터 유형을 변경합니다.
- 기본 테이블, 뷰 또는 별명에 주석을 추가 또는 변경합니다.

CONTROL

다음을 권한 부여합니다.

- 목록에 있는 적절한 모든 특권, 즉,
  - 기본 테이블에 ALTER, CONTROL, DELETE, INSERT, INDEX, REFERENCES, SELECT, UPDATE 등
  - 뷰에 CONTROL, DELETE, INSERT, SELECT, UPDATE 등
  - 별명에 대한 ALTER, CONTROL, INDEX 및 REFERENCES
- 위의 특권(CONTROL은 제외)을 다른 사람에게 권한 부여할 수 있는 능력.
- 기본 테이블, 뷰 또는 별명을 삭제할 수 있는 능력.

이 능력은 CONTROL 특권을 기초로 하여 다른 사용자에게 부여할 수 없습니다. 권한 부여할 수 있는 유일한 방법은 CONTROL 특권 자체를 부여하는 것이며, 이는 SYSADM 또는 DBADM 권한을 가진 사람만 수행할 수 있습니다.

- 테이블 및 색인에서 RUNSTATS 유틸리티를 실행하는 능력. RUNSTATS에 대해서는 *Command Reference*에서 정보를 참조하십시오.
- 기본 테이블 또는 요약 테이블에 대해 SET INTEGRITY문을 발행할 수 있는 능력.

기본 테이블, 요약 테이블 또는 별명의 정의자는 자동으로 CONTROL 특권을 받습니다.

뷰의 정의자는 fullselect에서 식별되는 모든 테이블, 뷰 및 별명의 CONTROL 특권을 보유하는 경우, 자동으로 CONTROL 특권을 받습니다.

DELETE

테이블이나 갱신 가능 뷰에서 행을 삭제하는 특권을 권한 부여합니다.

INDEX

테이블에 색인을 작성하거나 별명에 색인 스펙을 작성할 특권을 권한 부여합니다. 색인이나 색인 스펙 작성자는 자동으로 색인이나 색인 스펙에 대한 CONTROL 특권을 가집니다.(작성자에게 색인이나 색인 스펙을 삭제할 권한을 줍니다.) 또한, 작성자는 INDEX 특권이 권한 취소될 지라도 CONTROL 특권을 보유합니다.

INSERT

테이블이나 갱신 가능한 뷰로 행을 삽입하고 IMPORT 유틸리티를 실행할 특권을 권한 부여합니다.

REFERENCES

상위 테이블로 참조되는 외부 키를 작성 및 삭제하는 특권을 권한 부여합니다.

명령문의 권한 부여 ID가 다음 중 하나인 경우,

- DBADM 또는 SYSADM 특권
- 테이블에 대해 CONTROL 특권
- 테이블에 대해 REFERENCES WITH GRANT OPTION

권한 받은 사용자는 ALTER TABLE문을 통해 차후에 추가된 경우라도 테이블의 모든 컬럼을 상위 키로 사용하여 참조 제한조건을 작성할 수 있습니다. 그렇지 않으면, 권한 부여된 특권은 명령문의 권한 부여 ID가 식별된 테이블에 대해 갖는 권한 부여 가능한 컬럼 REFERENCES 특권입니다. 외부 키를 작성 또는 삭제하는 데 필요한 권한 부여에 대해 524 페이지의 『ALTER TABLE』에서 자세한 정보를 참조하십시오.

외부 키가 참조 별명에 정의될 수 없을 지라도 별명에 특권이 권한 부여될 수 있습니다.

REFERENCES (*column-name*,...)

컬럼 목록에 상위 키로 지정된 컬럼만을 사용하여 외부 키를 작성 및 삭제하기 위한 특권을 권한 부여합니다. 각 *column-name*은 ON절에 식별된 테이블의 컬럼을 식별하는 규정화되지 않은 이름이어야 합니다. 입력된 테이블, 입력된 뷰 또는 별명에는 컬럼 레벨의 REFERENCES 특권을 권한 부여할 수 없습니다(SQLSTATE 42997).

SELECT

다음 사항에 특권을 권한 부여합니다.

- 테이블이나 뷰로부터 행을 검색합니다.
- 테이블에 대한 뷰를 작성합니다.
- 테이블이나 뷰에 대해 EXPORT 유틸리티를 수행합니다. EXPORT에 대해서는 *Command Reference*에서 참조하십시오.

UPDATE

ON 절에서 식별된 테이블이나 갱신 가능 뷰에 UPDATE문을 사용할 수 있는 특권을 부여합니다.

명령문의 권한 부여 ID가 다음 중 하나인 경우,

- DBADM 또는 SYSADM 특권
- 테이블 또는 뷰에서의 CONTROL 특권
- 테이블 또는 뷰에서의 UPDATE WITH GRANT OPTION

권한 받은 사용자는 ALTER TABLE문을 사용하여 차후에 추가된 컬럼뿐 아니라 권한 준 사용자가 권한 부여 특권을 갖고 있는 테이블이나 뷰의 갱신 가

능한 모든 컬럼을 갱신할 수 있습니다. 그렇지 않으면, 권한 부여된 특권은 명령문 권한 부여 ID가 식별된 테이블이나 뷰에 대해 갖는 권한 부여 가능한 컬럼 UPDATE 특권입니다.

UPDATE (*column-name***,...)**

UPDATE문을 사용하여 컬럼 목록에 지정된 컬럼만을 갱신할 수 있는 특권을 권한 부여합니다. 각 *column-name*은 ON절에 식별된 테이블이나 뷰의 컬럼을 식별하는 규정화되지 않은 이름이어야 합니다. 입력된 테이블, 입력된 뷰 또는 별명에는 컬럼 레벨의 UPDATE 특권을 권한 부여할 수 없습니다 (SQLSTATE 42997).

ON TABLE *table-name* 또는 *view-name* 또는 *nickname*

특권이 권한 부여될 해당 테이블, 뷰 또는 별명을 지정합니다.

실행 불능 뷰 또는 실행 불능 요약 테이블에는 특권이 권한 부여되지 않습니다(SQLSTATE 51024). 선언된 임시 테이블에 대한 특권을 부여할 수 없는 경우도 있습니다(SQLSTATE 42995).

TO

특권이 권한 부여될 위치를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

하나 이상의 사용자 또는 그룹의 권한 부여 ID를 나열합니다.[99]

그룹에 권한 부여된 특권은 패키지의 DML문에 대한 권한 부여 점검에는 사용되지 않습니다. CREATE VIEW문을 처리하는 동안 기본 테이블에 관한 권한을 점검할 때에도 사용되지 않습니다.

DB2 Universal Database에서, 그룹에 권한 부여된 테이블 특권은 동적으로 준비된 명령문에만 적용됩니다. 예를 들어, PROJECT 테이블에 대

99. 명령문을 발행하는 사용자의 권한 부여 ID를 권한 부여하는 이전 버전의 제한사항이 제거되었습니다.

한 INSERT 특권이 그룹 D204에는 권한 부여되고 UBIQUITY(D204의 구성원)에는 부여되지 않았다면, UBIQUITY는 다음과 같은 명령문을 실행할 수 있습니다.

```
EXEC SQL EXECUTE IMMEDIATE :INSERT_STRING;
```

여기서, 문자열의 내용은 다음과 같습니다.

```
INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

그러나, 다음과 같은 명령문으로 프로그램을 사전 처리 컴파일하거나 바인드할 수 없습니다.

```
EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

PUBLIC

모든 사용자에게 특권을 권한 부여합니다.[100]

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사람에게 특권을 권한 부여할 수 있도록 합니다.

지정된 특권에 CONTROL이 포함되는 경우, WITH GRANT OPTION은 CONTROL을 제외한 적용 가능한 모든 특권에 적용됩니다(SQLSTATE 01516).

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  - *authorization-name*이 운영 시스템에 GROUP으로만 정의되어 있으면, GROUP으로 취급됩니다.
  - *authorization-name*이 운영 시스템에 USER로만 정의되어 있거나 정의되어 있지 않으면, USER로 취급됩니다.
  - *authorization-name*이 운영 시스템에서 둘다로 정의되거나 DCE 인증이 사용되면, 오류(SQLSTATE 56092)가 발생합니다.

100. 정적 SQL문과 CREATE VIEW문에 대해 PUBLIC에 권한 부여된 특권을 사용하는 이전 버전의 제한사항이 제거되었습니다.

- 일반적으로, GRANT문은 명령문의 권한 부여 ID가 권한 부여하도록 허용된 특권 권한 부여 프로세스를 수행하며, 두 개 이상의 특권이 권한 부여되지 않은 경우 경고(SQLSTATE 01007)를 리턴합니다. 아무 특권도 권한 부여되지 않으면, 오류가 리턴됩니다(SQLSTATE 42501).[101] CONTROL 특권이 지정되면, 명령문의 권한 부여 ID에 SYSADM 또는 DBADM 권한이 있는 경우에만 특권이 부여됩니다(SQLSTATE 42501).

주

- 테이블 계층의 모든 레벨에서 특권이 독립적으로 권한 부여될 수 있습니다. 상위 테이블에 대한 특권을 가진 사용자는 서브테이블에 영향을 줄 수 있습니다. 예를 들어, 상위 테이블 *T*를 지정하는 갱신은, *T*에 대해 UPDATE 특권을 가지고 있지만 *S*에 대해서는 UPDATE 특권이 없는 사용자에 의해 *T*의 서브테이블 *S*에 있는 행에 대한 변경으로서 나타날 수 있습니다. 사용자는 필요한 특권이 서브테이블에 있는 경우에만 그 서브테이블을 직접 조작할 수 있습니다.
- 별명 특권을 권한 부여하면 데이터 소스 오브젝트(테이블이나 뷰) 특권에 아무 영향도 미치지 않습니다. 보통, 데이터 소스 특권은 데이터 검색을 시도할 때 별명이 참조하는 테이블이나 뷰를 위해 필요합니다.
- 별명에 관한 조작은 별명을 참조하는 명령문이 처리될 때 데이터 소스에서 사용된 권한 부여 ID의 특권에 의존하기 때문에, 별명에는 DELETE, INSERT, SELECT 및 UPDATE 특권이 정의되지 않습니다.

예

예 1: WESTERN_CR 테이블에 대한 모든 특권을 PUBLIC으로 부여하십시오..

```
GRANT ALL ON WESTERN_CR
 TO PUBLIC
```

예 2: 사용자 PHIL과 CLAIRE가 CALENDAR 테이블을 읽고 새로운 항목을 삽입할 수 있도록 CALENDAR 테이블에 대한 적절한 특권을 권한 부여합니다. 기존의 항목을 변경하거나 제거하지 못하도록 합니다.

101. 명령문 처리에 사용된 패키지가 SQL92E 또는 MIA에 설정된 LANGLEVEL로 사전 처리 컴파일되었으면, 권한 준 사용자 권한 부여 오브젝트에 대한 특권이 없는한 경고가 리턴됩니다(SQLSTATE 01007).

```
GRANT SELECT, INSERT ON CALENDAR
 TO USER  PHIL, USER CLAIRE
```

예 3: COUNCIL 테이블의 모든 특권을 사용자 FRANK에게 권한 부여하고 모든 특권을 다른 사람에게로 확대할 수 있는 능력을 부여합니다.

```
GRANT ALL ON COUNCIL
 TO USER FRANK WITH GRANT OPTION
```

예 4: 테이블 CORPDATA.EMPLOYEE에 대한 SELECT 특권을 JOHN이라는 사용자에게 권한 부여합니다. JOHN이라는 사용자는 있고 그룹은 없습니다.

```
GRANT SELECT ON CORPDATA.EMPLOYEE
TO JOHN
```

또는

```
GRANT SELECT
  ON CORPDATA.EMPLOYEE TO USER JOHN
```

예 5: 테이블 CORPDATA.EMPLOYEE에 대한 SELECT 특권을 JOHN이라는 그룹에 권한 부여합니다. JOHN이라는 그룹은 있고 사용자는 없습니다.

```
GRANT SELECT ON CORPDATA.EMPLOYEE
TO JOHN
```

또는

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO GROUP JOHN
```

예 6: 테이블 T1에 대한 INSERT와 SELECT 특권을 그룹 D024와 사용자 D024에게 권한 부여합니다.

```
GRANT INSERT, SELECT ON TABLE T1
   TO GROUP D024, USER D024
```

이런 경우, D024 그룹 구성원과 D024 사용자 모두는 테이블 T1로 INSERT 및 SELECT할 수 있습니다. 또한, SYSCAT.TABAUTH 카탈로그 뷰에 추가되는 두 개의 행이 있습니다.

예 7: CALENDAR 테이블에 대한 INSERT, SELECT 및 CONTROL 특권을 사용자 FRANK에게 권한 부여합니다. FRANK는 특권을 다른 사용자에게 전달할 수 있어야 합니다.

```
GRANT CONTROL ON TABLE CALENDAR
   TO FRANK WITH GRANT OPTION
```

이 명령문의 실행 결과 CONTROL에 WITH GRANT OPTION이 제공되지 않았다는 경고(SQLSTATE 01516)가 리턴됩니다. Frank는 필요한 대로 INSERT와 SELECT 등 CALENDAR에 관한 모든 특권을 권한 부여할 수 있는 능력을 갖게 됩니다. FRANK는 SYSADM이나 DBADM 권한을 가지고 있는 경우가 아니면 CALENDAR에 대한 CONTROL을 다른 사용자에게 권한 부여할 수 없습니다.

예 8: 사용자 JON이 색인이 없는 Oracle 테이블에 대한 별명(nickname)을 작성했습니다. 별명은 ORAREM1입니다. 나중에, Oracle DBA가 이 테이블에 대한 색인을 정의했습니다. 사용자 SHAWN이 이제 DB2가 이 색인이 존재함을 알아, 최적화 알고리즘이 테이블을 보다 효율적으로 액세스할 수 있는 전략을 고안할 수 있기를 원합니다. SHAWN은 ORAREM1에 대한 색인 스펙을 작성함으로써 DB2에 색인을 알릴 수 있습니다. SHAWN에게 이 별명에 대한 색인 특권을 부여하여, 색인 스펙을 작성할 수 있게 하십시오.

```
GRANT INDEX ON NICKNAME ORAREM1
   TO USER SHAWN
```

GRANT(테이블 공간 권한 취소)

이러한 형태의 GRANT문은 테이블에 대한 특권을 부여합니다.

호출

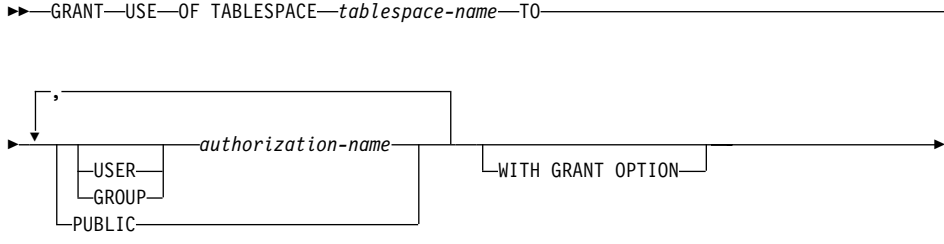
이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- 테이블 공간 사용에 대한 WITH GRANT OPTION
- SYSADM, SYSCTRL 또는 DBADM 권한

구문



설명

USE

테이블 작성시 테이블 공간을 지정하거나 기본값으로 설정할 수 있는 특권을 부여합니다. 테이블 공간의 작성자는 권한 부여 옵션을 사용하여 USE 특권을 자동으로 받습니다.

OF TABLESPACE *tablespace-name*

USE 특권이 부여될 테이블 공간을 식별합니다. 테이블 공간은

SYSCATSPACE(SQLSTATE 42838) 또는 SYSTEM TEMPORARY 테이블 공간(SQLSTATE 42809)일 수 없습니다.

TO

USE 특권이 부여되는 사용자를 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name

여러 사용자 또는 그룹의 권한 부여 ID를 나열합니다.

권한 부여 ID 목록에는 명령문을 발행하는 사용자의 권한 부여 ID가 포함될 수 없습니다(SQLSTATE 42502).

PUBLIC

모든 사용자에게 USE 특권을 권한 부여합니다.

WITH GRANT OPTION

지정된 *authorization-name*이 다른 사람에게 특권을 권한 부여할 수 있도록 합니다.

WITH GRANT OPTION가 생략되면, 지정된 *authorization-name*은 다음의 경우에만 USE 특권을 GRANT할 수 있습니다.

- SYSADM 또는 DBADM 권한이 있는 경우
- 다른 소스의 USE 특권을 GRANT하기 위한 능력을 제공받은 경우

주

USER와 GROUP이 모두 지정되지 않은 경우,

- *authorization-name*이 운영 시스템에 GROUP으로만 정의되어 있으면, GROUP으로 취급됩니다.
- *authorization-name*이 운영 시스템에 USER로만 정의되어 있거나 정의되어 있지 않으면, USER로 취급됩니다.

- *authorization-name*이 운영 시스템에서 둘다로 정의되거나 DCE 인증이 사용되면, 오류(SQLSTATE 56092)가 발생합니다.

예

예 1: 사용자 BOBBY에게 테이블 공간 PLANS에 테이블을 작성하고 이 특권을 다른 사용자에게 부여할 수 있는 능력을 부여하십시오.

```
GRANT USE OF TABLESPACE PLANS TO BOBBY WITH GRANT OPTION
```

INCLUDE

INCLUDE문은 선언을 원시 프로그램에 삽입합니다.

호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 실행되지 않습니다.

권한 부여

필요 사항 없음.

구문

설명

SQLCA

SQL 통신 영역(SQLCA)의 설명이 포함되도록 표시합니다. SQLCA 설명에 관해서는 1261 페이지의『부록B. SQL 통신(SQLCA)』에서 참조하십시오.

SQLDA

SQL 설명자 영역(SQLDA)의 설명이 포함되도록 표시합니다. SQLDA 설명에 관해서는 1267 페이지의『부록C. SQL 설명자 영역(SQLDA)』에서 참조하십시오.

이름

사전 처리 컴파일중인 원시 프로그램에 포함될 원문이 들어 있는 외부 파일을 식별합니다. 파일 이름 확장자가 없는 SQL 식별자일 수도 있고, 작은 따옴표(' ')로 표시되는 리터럴일 수도 있습니다. SQL 식별자는 사전 처리 컴파일되는 원시 파일의 파일 확장자 이름을 취합니다. 따옴표로 표시되는 리터럴이 파일 이름 확장자를 제공하지 않는 경우, 아무 것도 취하지 않습니다.

호스트 언어 특정 정보에 대해서는 *응용프로그램 개발 안내서*에서 참조하십시오.

주

- 프로그램이 사전 처리 컴파일될 때, INCLUDE문은 원시 명령문으로 대체됩니다. 따라서, INCLUDE문은 결과 원시 명령문은 컴파일러에서 허용할 수 있는 시점에 지정되어야 합니다.
- 외부 원시 파일은 *이름*에 의해 지정되는 호스트 언어로 작성되어야 합니다. 18자 이상이 되거나, SQL 식별자에 허용되지 않는 문자가 들어 있는 경우, 작은 따옴표로 묶여야 합니다. INCLUDE *이름* 명령문은 순환되지는 않으나 중첩될 수 있습니다.(예를 들어, A와 B가 모듈이고 A에 INCLUDE *이름* 명령문이 들어 있는 경우, A가 B를 호출한 후 B가 A를 호출하는 것은 유효하지 않습니다.)
- LANGLEVEL 사전 처리 컴파일 옵션이 SQL92E 값으로 지정될 때에는 INCLUDE SQLCA가 지정되면 안 됩니다. SQLSTATE와 SQLCODE 변수를 호스트 변수 선언 섹션 내에서 정의할 수 있습니다.

예

C 프로그램에 SQLCA를 포함시킵니다.

```
 EXEC SQL INCLUDE SQLCA;
EXEC SQL  DECLARE C1 CURSOR FOR
   SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
    WHERE ADMRDEPT = 'A00';
EXEC SQL  OPEN C1;
 while (SQLCODE==0) {
   EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
 (Print results)
 }
EXEC SQL  CLOSE C1;
```

INSERT

INSERT문은 테이블 또는 뷰에 행을 삽입합니다. 뷰에 행을 삽입하면 그 뷰의 기본이 되는 테이블에도 행이 삽입됩니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문을 실행하려면, 그 명령문의 권한 부여 ID에서 갖고 있는 특권에 적어도 다음 중 하나가 포함되어야 합니다.

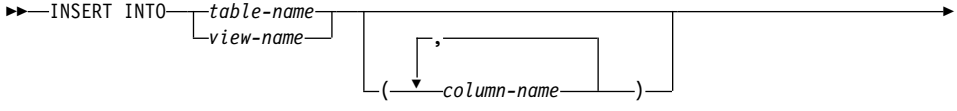
- 행이 삽입될 뷰 또는 테이블에서의 INSERT 특권
- 행이 삽입될 뷰 또는 테이블에서의 CONTROL 특권
- SYSADM 또는 DBADM 권한

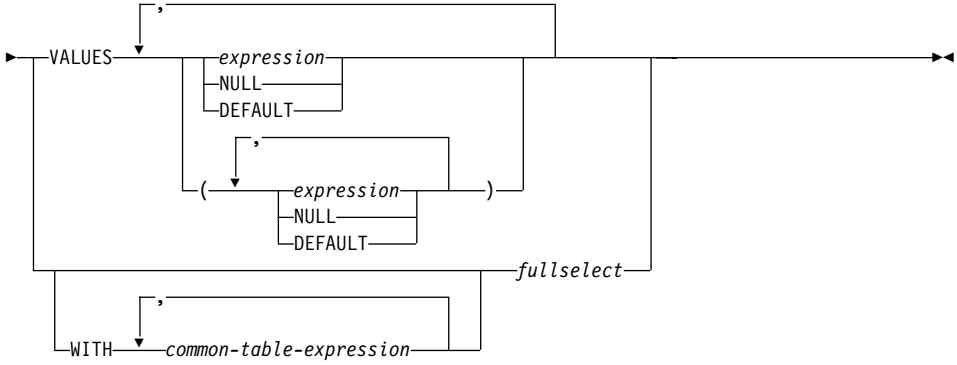
또한, INSERT문에서 사용되는 fullselect에 참조되는 각 테이블 또는 뷰마다, 명령문의 권한 부여 ID가 보유하는 특권에는 최소한 다음 중 하나가 포함되어야 합니다.

- SELECT 특권
- CONTROL 특권
- SYSADM 또는 DBADM 권한

정적 INSERT문에 대해서는 GROUP 특권이 점검되지 않습니다.

구문





주: *common-table-expression* 및 *fullselect*의 구문에 대해서는 433 페이지의 『제 5장 조회』에서 참조하십시오.

설명

INTO *table-name* 또는 *view-name*

삽입 조작의 오브젝트를 식별합니다. 이름은 응용프로그램 서버(AS)에 있는 테이블 또는 뷰를 식별해야 하나 카탈로그 테이블, 요약 테이블, 카탈로그 테이블 뷰 또는 읽기 전용 뷰를 식별해서는 안 됩니다.

다음에서 파생된 뷰 컬럼에는 값이 들어갈 수 없습니다.

- 상수, 표현식 또는 스칼라 함수
- 다른 뷰 컬럼과 같은 기본 테이블 컬럼
- 별명으로부터 파생된 컬럼.

삽입 조작의 오브젝트가 그러한 컬럼을 가진 뷰인 경우, 컬럼 이름 목록이 지정되어야 하고, 그 목록은 이들 컬럼을 식별해서는 안 됩니다.

(*column-name***,...)**

삽입 값이 제공될 컬럼을 지정합니다. 각 설명은 테이블이나 뷰의 컬럼을 식별하는 자격 요건이 안되는 이름이어야 합니다. 같은 컬럼은 한 번 이상 식별될 수 없습니다. 삽입 값을 허용할 수 없는 뷰 컬럼을 식별해서는 안 됩니다.

컬럼 목록을 생략하면, 테이블 또는 뷰의 모든 컬럼이 왼쪽에서 오른쪽으로 식별되도록 목록이 암시적으로 지정됩니다. 이 목록은 명령문이 준비되고, 따라서 명령문이 준비된 이후 테이블에 추가된 컬럼을 포함하지 않을 때 만들어집니다.

내재된 컬럼 목록은 준비시에 형성됩니다. 그러므로 응용프로그램 프로그램에 내포된 INSERT문은 준비 후 테이블이나 뷰에 추가되었을 어떠한 컬럼도 사용하지 않습니다.

VALUES

삽입될 하나 이상의 값 행을 소개합니다.

명명된 각 호스트 변수는 호스트 변수 선언 규칙에 따라 프로그램에서 기술되어야 합니다.

각 행에 대한 값의 수는 컬럼 목록의 이름 수와 같아야 합니다. 첫번째 값은 목록의 첫번째 컬럼에, 두 번째 값은 두 번째 컬럼에 삽입됩니다.

expression

*expression*은 182 페이지의 『표현식』처럼 기술될 수 있습니다.

NULL

널(NULL) 입력 가능 컬럼에 대해서만 지정되어야 하는 널(NULL) 값을 지정합니다.

DEFAULT

기본값이 사용되도록 지정합니다. DEFAULT 지정 결과는 다음과 같이 컬럼 정의 방법에 따라 달라집니다.

- 표현식을 기초로 컬럼이 생성된 컬럼으로 정의된 경우, 시스템이 표현식을 근거로 컬럼 값을 생성합니다.
- IDENTITY절이 사용되면 데이터베이스 관리 프로그램에서 값을 생성합니다.
- WITH DEFAULT절이 사용되면, 삽입된 값이 컬럼에 대해 정의됩니다 (800 페이지의 『CREATE TABLE』의 *default-clause* 참조).
- WITH DEFAULT절, GENERATED절 및 NOT NULL절이 사용되지 않을 경우, 삽입된 값은 NULL입니다.

- NOT NULL절이 사용되고 GENERATED절이 사용되지 않거나 WITH DEFAULT절이 사용되지 않거나 DEFAULT NULL이 사용되면, 해당 컬럼에 대해 DEFAULT 키워드를 지정할 수 없습니다(SQLSTATE 23502).

WITH *common-table-expression*

뒤에 나올 fullselect과 함께 사용할 공통 테이블 표현식을 정의합니다. *common-table-expression*에 대한 483 페이지의 『공통 테이블 표현식』에서 내용을 참조하십시오.

fullselect

fullselect의 결과 테이블 형태로 새로운 행 세트를 지정합니다. 하나 또는 그 이상 또는 하나도 없을 수 있습니다. 결과 테이블이 비어 있는 경우, SQLCODE는 +100으로 설정되고 SQLSTATE는 '02000'으로 설정됩니다.

INSERT의 기본 오브젝트와 fullselect의 기본 오브젝트 또는 fullselect의 부속 조회가 같은 테이블인 경우, 행이 삽입되기 전에 fullselect가 완전히 평가됩니다.

결과 테이블의 컬럼 수는 컬럼 목록의 이름 수와 같아야 합니다. 결과의 첫째 컬럼 값은 목록의 첫째 컬럼에, 두 번째 값은 둘째 컬럼에, 이와 같은 식으로 삽입됩니다.

규칙

- ***기본값:*** 컬럼 목록에 없는 컬럼에 삽입된 값은 컬럼의 기본값이거나 널(NULL)입니다. 널(NULL) 값을 허용하지 않고 NOT NULL WITH DEFAULT로 정의되지 않은 컬럼은 컬럼 목록에 포함되어야 합니다. 마찬가지로, 뷰에 삽입하는 경우, 뷰에 없는 기본 테이블의 컬럼에 삽입되는 값은 컬럼의 기본값 또는 널(NULL)입니다. 그러므로, 뷰에 없는 기본 테이블의 모든 컬럼에는 기본값이 있거나 널(NULL) 값을 허용해야 합니다. GENERATED ALWAYS절로 정의된 생성된 컬럼에 삽입할 수 있는 유일한 값은 DEFAULT입니다(SQLSTATE 428C9).
- ***길이:*** 컬럼의 삽입 값이 숫자라면, 컬럼은 숫자의 정수 부분을 나타내는 숫자 컬럼이어야 합니다. 컬럼의 삽입 값이 문자열인 경우, 컬럼은 적어도 그 문자열 길

이의 길이 속성을 지닌 문자열 컬럼이 되거나, 문자열이 날짜, 시간 또는 시간 소인을 나타내는 경우, 날짜 시간 컬럼이 됩니다.

- ***지정:*** 『제3장 언어 요소』에 기술된 지정 규칙에 따라 삽입 값이 컬럼에 지정됩니다.
- ***유효성:*** 명명된 테이블 또는 명명된 뷰의 기본 테이블에 하나 이상의 고유 색인이 있는 경우, 테이블에 삽입된 각 행은 이러한 색인으로 강요되는 제한조건에 따라야 합니다. 정의에 WITH CHECK OPTION이 포함된 뷰가 명명되면 그 뷰에 삽입된 각 행은 뷰의 정의에 따라야 합니다. 이러한 상황에 적용되는 규칙에 대한 설명을 보려면 931 페이지의 『CREATE VIEW』에서 참조하십시오.
- ***참조 무결성:*** 테이블에 정의된 각 제한조건의 경우, 외부 키의 널(NULL)이 아닌 삽입 값 각각은 상위 테이블의 기본 키 값과 같아야 합니다.
- ***점검 제한조건:*** 삽입 값은 테이블에 정의된 점검 제한조건의 점검 조건을 만족시켜야 합니다. 정의된 점검 제한조건을 가진 테이블에 INSERT하는 경우, 삽입되는 각 행에 대해 의무 규정 조건이 한 번 평가됩니다.
- ***트리거:*** INSERT문은 트리거를 실행시킵니다. 트리거로 인해 다른 명령문이 실행될 수도 있고, 삽입 값에 기초하여 오류가 발생할 수도 있습니다.
- ***데이터링크:*** DATALINK 값을 포함하는 INSERT문은 URL 값이 포함되고(빈 문자열이나 공백이 아님) FILE LINK CONTROL로 컬럼이 정의된 경우 파일의 링크를 시도합니다. DATALINK 값이나 파일 링크 오류가 발생하면 삽입이 실패하게 됩니다(SQLSTATE 428D1 또는 57050).

주

- 프로그램 내에 내포된 INSERT문이 실행된 후, SQLCA의 SQLERRD(3) 부분의 세번째 변수 값은 삽입된 행의 수를 표시합니다. SQLERRD(5)에는 모든 트리거된 삽입, 갱신 및 삭제 조작 수의 합계가 들어 있습니다.
- 적절하게 잠금이 되어 있지 않는 한, 성공적인 INSERT문의 실행시 하나 이상의 독점적인 잠금이 생깁니다. 잠금이 해제될 때까지 삽입된 행은 다음에 의해서만 액세스될 수 있습니다.
  - 삽입을 수행하는 응용프로그램 프로세스.

  - 읽기 전용 커서를 통해 분리 레벨 UR을 사용하는 다른 응용프로그램 프로세스, SELECT INTO문 또는 부속 조회에 사용되는 부속 선택.

- 잠금에 대해 자세히 알려면, COMMIT, ROLLBACK 및 LOCK TABLE문에 대한 설명을 보십시오.
- 파티션된 데이터베이스에 대해 응용프로그램이 실행되고, INSERT BUF 옵션으로 바운드된 경우, EXECUTE IMMEDIATE를 사용하여 처리되지 않은 INSERT with VALUES문은 버퍼됩니다. DB2가 이러한 INSERT문을 응용프로그램 로직의 루프 내에서 처리합니다. 명령문이 완료에 실행되기 보다는 하나 이상의 버퍼에 새 행 값을 버퍼하려고 합니다. 결국 행들의 테이블의 실제 삽입은 응용프로그램의 INSERT 로직과 다른 시점, 즉 보다 나중에 수행됩니다. 이러한 비동기 삽입은 INSERT와 관련된 오류가 응용프로그램에서 INSERT에 후속하는 다른 SQL문에 리턴되게 할 수 있습니다.

  이것은 INSERT의 성능을 상당히 향상시킬 잠재력을 가지고 있지만, 오류 처리의 비동기성 때문에 깨끗한 데이터와 함께 사용하는 것이 가장 좋습니다. 버퍼 삽입에 대해서는 *응용프로그램 개발 안내서*에서 참조하십시오.
- 식별 컬럼이 있는 테이블에 행이 삽입될 경우 DB2는 해당 식별 컬럼에 대한 값을 생성합니다.
  - GENERATED ALWAYS 식별 컬럼의 경우는 DB2가 항상 값을 생성합니다.
  - GENERATED BY DEFAULT 컬럼의 경우, 값이 명시적으로 지정되지 않았다면(VALUES절이나 부속 선택을 사용하여) DB2가 값을 생성합니다.

  DB2가 생성하는 첫번째 값은 식별 컬럼에 대한 START WITH 스펙의 값입니다.
- 사용자 정의 구별 유형 식별 컬럼에 대한 값이 삽입되는 경우, 전체 계산이 소스 유형으로 수행되고 그 값이 실제로 컬럼에 지정되기 전에 결과가 구별 유형으로 변환됩니다.[102]
- GENERATED ALWAYS 식별 컬럼으로 삽입될 때, DB2는 항상 해당 컬럼에 대한 값을 생성하고 사용자는 삽입시 값을 지정할 수 없습니다.

---

102. 이전 값은 계산에 앞서 소스 유형으로 변환되지 않습니다.

GENERATED ALWAYS 식별 컬럼이 VALUES절의 기본값이 아닌 다른 값으로 INSERT문의 컬럼 목록에 나열되면 오류가 발생합니다(SQLSTATE 428C9).

예를 들어, EMPID가 GENERATED ALWAYS인 식별 컬럼으로 정의되었다고 가정할 때 다음 명령은

```
INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
  VALUES (:hv_valid_emp_id, :hv_name, :hv_addr)
```

오류를 발생시킵니다.

- GENERATED BY DEFAULT 컬럼에 삽입될 때, DB2는 컬럼의 실제 값이 VALUES절내에서 또는 부속 선택으로부터 해당 컬럼의 실제 값이 지정되게 합니다. 그러나 VALUES절에 값이 지정되면 DB2가 값의 검증을 수행하지 않습니다. 값의 고유성을 보증하기 위해서는 식별 컬럼의 고유 색인을 작성해야 합니다.

  컬럼 목록을 지정하지 않고 GENERATED BY DEFAULT 식별 컬럼이 있는 테이블에 삽입할 때, VALUES절은 DEFAULT 키워드를 지정하여 식별 컬럼의 값을 나타낼 수 있습니다. DB2는 해당 식별 컬럼의 값을 생성합니다.

  ```
  INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
    VALUES (DEFAULT, :hv_name, :hv_addr)
  ```

  이 예에서 EMPID는 식별 컬럼으로 정의되므로, DB2가 이 컬럼에 삽입된 값을 생성합니다.

- 부속 선택으로 식별 컬럼에 삽입 규칙은 VALUES절로 삽입하는 규칙과 유사합니다. 식별 컬럼이 GENERATED BY DEFAULT로 정의된 경우에만 식별 컬럼의 값을 지정할 수 있습니다.

  예를 들어, T1 및 T2가 *intcol1*과 *identcol2* 컬럼 모두를 포함하는 동일한 정의를 갖는 테이블입니다(두 컬럼 모두는 INTEGER 유형이고 두 번째 컬럼은 식별 속성을 가집니다). 다음 삽입을 고려하십시오.

  ```
   INSERT INTO T2
  SELECT *
     FROM T1
  ```

  이 예는 논리적으로 다음과 동등합니다.

```
INSERT INTO T2 (intcol1,identcol2)
   SELECT intcol1, identcol2
   FROM T1
```

두 경우 모두에서 INSERT문은 T2의 식별 컬럼에 대한 명시적 값을 제공합니다. 이 명시적 스펙에는 식별 컬럼의 값을 제공할 수 있으나, T2의 식별 컬럼을 GENERATED BY DEFAULT로 정의해야 합니다. 그렇지 않으면, 오류가 발생합니다(SQLSTATE 428C9).

테이블에 GENERATED ALWAYS 식별로 정의된 컬럼이 있는 경우, 여전히 같은 정의를 갖는 테이블로부터 다른 모든 컬럼을 전파할 수 있습니다. 예를 들어, 위에 기술된 예제 테이블 T1 및 T2가 제공된 경우 다음 SQL을 사용하여 intcol1 값을 T1에서 T2로 전파할 수 있습니다.

```
INSERT INTO T2 (intcol1)
   SELECT intcol1
   FROM T1
```

identcol2가 컬럼 목록에 지정되지 않았기 때문에, 이는 기본(생성된) 값으로 채워집니다.

- 컬럼이 GENERATED ALWAYS 식별 컬럼으로 정의된 단일 컬럼 테이블에 행을 삽입하는 경우, DEFAULT 키워드로 VALUES절을 지정할 수 있습니다. 이 경우 응용프로그램은 테이블에 대한 어떤 값도 제공하지 않으며 DB2가 해당 식별 컬럼의 값을 생성합니다.

  ```
  INSERT INTO IDTABLE
     VALUES(DEFAULT)
  ```

  컬럼이 식별 속성을 갖는 동일한 단일 컬럼 테이블의 경우, 단일 INSERT문으로 여러 행을 삽입하려면 다음 INSERT문을 사용할 수 있습니다.

  ```
  INSERT INTO IDTABLE
     VALUES (DEFAULT), (DEFAULT), (DEFAULT), (DEFAULT)
  ```

- DB2가 식별 컬럼의 값을 생성할 때 생성된 값이 소모됩니다. 다음 번에 값이 필요하면 DB2가 새 값을 생성합니다. 이는 식별 컬럼을 포함하는 INSERT문이 실패하거나 구간 복원되는 경우에도 적용됩니다.

  예를 들어, 식별 컬럼에 대한 고유 색인이 작성되었다고 가정합니다. 식별 컬럼의 값 생성시 중복되는 키 위반이 발견되면 오류가 발생하고(SQLSTATE 23505)

해당 식별 컬럼에 대해 생성된 값이 소모된 것으로 간주됩니다. 이는 식별 컬럼이 GENERATED BY DEFAULT로 정의되었고 시스템이 새 값을 생성하려 하나 사용자가 이전 INSERT문에 식별 컬럼 값을 명시적으로 지정한 경우 발생합니다. 이 경우 동일한 INSERT문을 다시 발행하면 성공할 수 있습니다. DB2가 식별 컬럼에 대한 다음 값을 생성하게 되고 이러한 다음 값이 고유하게 되며 이 INSERT문이 성공할 수 있습니다.

- 식별 컬럼 값을 생성할 때 식별 컬럼의 최대값(또는 내림차순으로 최소값)이 초과되면 오류가 발생합니다(SQLSTATE 23522). 이러한 상황에서 사용자는 범위가 더 큰 식별 컬럼이 있는 새 테이블을 삭제하고 작성해야 합니다(즉, 더 큰 값 범위가 가능하도록 데이터 유형을 변경하거나 컬럼 값을 증가시키십시오).

  예를 들어, 데이터 유형이 SMALLINT인 식별 컬럼을 정의했을 수 있으며 결과적으로 해당 컬럼은 지정 가능한 값을 모두 소모합니다. 식별 컬럼을 INTEGER로 다시 정의하려면, 데이터를 로드 해제해야 하고 테이블을 삭제한 다음 해당 컬럼에 대한 새 정의로 다시 작성하고 데이터를 다시 로드해야 합니다. 테이블이 다시 정의되는 경우, DB2가 생성하는 다음 값이 원래 순서의 다음 값이 되도록 식별 컬럼에 대해 START WITH값을 지정해야 합니다. 끝 값을 결정하려면, 데이터를 로드 해제하기 전에 식별 컬럼의 MAX(오름차순의 경우) 또는 식별 컬럼의 MIN(내림차순의 경우)을 사용하여 조회를 발행하십시오.

예

예 1: 다음 스펙으로 DEPARTMENT 테이블에 새로운 부서를 삽입합니다.

- 부서 번호(DEPTNO)는 'E31'입니다.
- 부서 이름(DEPTNAME)은 'ARCHITECTURE'입니다.
- 번호가 '00390'인 사람에 의해 관리됩니다(MGRNO).
- 'E01' 부서로 보고합니다(ADMRDEPT).

```
INSERT INTO DEPARTMENT
 VALUES ('E31', 'ARCHITECTURE', '00390', 'E01')
```

예 2: 예 1에서와 같이 DEPARTMENT 테이블에 새로운 부서를 삽입하지만 새로운 부서에 관리자를 지정하지는 않습니다.

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
   VALUES ('E31', 'ARCHITECTURE', 'E01')
```

예 3: 예 2에서와 같이 하나의 명령문을 사용하여 두 개의 새로운 부서를 DEPARTMENT 테이블에 삽입하지만 새로운 부서에 관리자를 지정하지는 않습니다.

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
   VALUES ('B11', 'PURCHASING', 'B01'),
          ('E41', 'DATABASE ADMINISTRATION', 'E01')
```

예 4: EMP_ACT 테이블과 같은 컬럼을 갖는 임시 테이블 MA_EMP_ACT를 작성하십시오. 문자 'MA'로 시작하는 프로젝트 번호(PROJNO)를 갖는 EMP_ACT 테이블의 행으로 MA_EMP_ACT를 로드하십시오.

```
CREATE TABLE MA_EMP_ACT
      ( EMPNO CHAR(6)  NOT NULL,
                              PROJNO CHAR(6)  NOT NULL,
                              ACTNO SMALLINT  NOT NULL,
                              EMPTIME DEC(5,2),
                                    EMSTDATE DATE,
                              EMENDATE  DATE )
INSERT INTO MA_EMP_ACT
   SELECT * FROM EMP_ACT
     WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

예 5: C 프로그램 명령문을 사용하여 스켈리톤 프로젝트를 PROJECT 테이블에 추가합니다. 호스트 변수로부터 프로젝트 번호(PROJNO), 프로젝트 이름(PROJNAME), 부서 번호(DEPTNO), 그리고 담당 직원(RESPEMP)을 알아내십시오. 현재 날짜를 프로젝트 시작일(PRSTDATE)로 사용합니다. NULL 값을 테이블에 남아있는 컬럼에 지정합니다.

```
EXEC SQL  INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTDATE )
            VALUES (:PRJNO, :PRJNM, :DPTNO, :REMP, CURRENT DATE);
```

LOCK TABLE

LOCK TABLE문은 응용프로그램 프로세스에서 동시에 테이블을 변경하지 못하도록 하거나, 응용프로그램 프로세스에서 동시에 테이블을 사용하지 못하도록 합니다.

호출

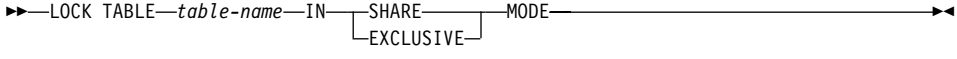
이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- 테이블의 SELECT 특권
- 테이블에 대해 CONTROL 특권
- SYSADM 또는 DBADM 권한

구문



설명

table-name

테이블을 식별합니다. *table-name*은 응용프로그램 서버(AS)에 있는 테이블을 식별해야 하며, 카탈로그 테이블을 식별해서는 안 됩니다. 별명(SQLSTATE 42809)이나 선언된 임시 테이블(SQLSTATE 42995)이 될 수는 없습니다. *table-name*이 입력된 테이블인 경우, 테이블 계층의 루트 테이블이어야 합니다(SQLSTATE 428DR).

IN SHARE MODE

동시에 응용프로그램 프로세스에서 테이블을 읽을 수는 있으나 다른 조작은 실행하지 못하도록 합니다.

IN EXCLUSIVE MODE

동시에 응용프로그램 프로세스에서 테이블을 조작하지 못하도록 합니다. EXCLUSIVE MODE는 분리 레벨 미확약 읽기(UR)에서 실행중인 응용프로그램 프로세스에서 동시에 테이블을 읽을 수 있도록 합니다.

주

- 동시적인 조작을 못하도록 잠금이 사용됩니다. 잠금은 적절한 잠금이 이미 존재하는 경우, LOCK TABLE문의 실행중에 반드시 얻게 되는 것은 아닙니다. 동시적인 조작을 막는 잠금은 최소한 작업 단위(UOW)가 종료될 때까지는 그대로 있습니다.
- 파티션된 데이터베이스에서, 노드 그룹의 첫번째 파티션(가장 낮은 번호를 갖는 파티션)에서 테이블 잠금이 가장 먼저 이루어지고 그 다음에 또 다른 파티션에서 잠금이 이루어집니다. LOCK TABLE문이 인터럽트될 경우, 테이블이 일부 파티션에서는 잠기지만 다른 파티션에서는 잠기지 않을 수 있습니다. 이 경우, 또 다른 LOCK TABLE문을 실행하여 모든 파티션에서 잠금을 완료하거나 COMMIT 또는 ROLLBACK문을 실행하여 현재의 잠금을 해제하십시오.
- 이 명령문은 노드 그룹의 모든 파티션에 영향을 줍니다.

예

테이블 EMP상에서 잠금을 수행합니다. 다른 프로그램이 테이블을 읽거나 갱신하지 못하도록 합니다.

```
LOCK TABLE EMP IN EXCLUSIVE MODE
```

OPEN

OPEN문은 커서를 열어 그 결과 테이블의 행을 페치하는 데 사용됩니다.

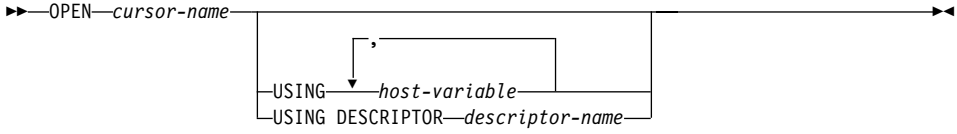
호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

커서를 사용할 때 필요한 권한에 대해서는 952 페이지의 『DECLARE CURSOR』에서 참조하십시오.

구문



설명

cursor-name

프로그램에서 이미 명시된 DECLARE CURSOR문에 정의된 커서를 명명합니다. OPEN문이 실행될 때 커서는 잠긴 상태에 있어야 합니다.

DECLARE CURSOR문은 다음과 같은 방식으로 SELECT문을 식별해야 합니다.

- DECLARE CURSOR문에 SELECT문 포함
- 준비된 SELECT문을 명명하는 *명령문 이름* 포함

커서의 결과 테이블은 SELECT문을 평가하고, 그 안에 있거나 OPEN문의 USING절에 있는 지정된 호스트 변수의 현재 값을 사용하여 유도됩니다. 결과 테이블의 행은 OPEN문 실행중에 도출되고, 임시 테이블은 그 행을 보유하기 위해 작성됩니다. 또는, 후속 FETCH문 실행 중에 도출될 수도 있습니

다. 어떤 경우이든, 커서는 열린 상태에 놓이며, 결과 테이블의 첫째 행 앞에 위치합니다. 테이블이 빈 경우, 커서 상태는 "마지막 행 다음"에서 효력을 발생합니다.

USING

준비된 명령문의 매개변수 표시문자 대신에 사용되는 값인 호스트 변수의 목록이 표시됩니다. (매개변수 표시문자에 대해 1087 페이지의 『PREPARE』에서 참조하십시오.) DECLARE CURSOR문이 매개변수 표시문자가 포함된 준비된 명령문을 명명하는 경우, USING을 사용해야 합니다. 준비된 명령문에 매개변수 표시문자가 포함되지 않는 경우, USING이 무시됩니다.

host-variable

호스트 변수 선언 규칙에 따라 프로그램에 기술된 변수를 식별합니다. 변수의 수는 준비된 명령문의 매개변수 표시문자 수와 같아야 합니다. *n*번째 변수는 준비된 명령문의 *n*번째 매개변수에 일치합니다. 위치 지정자 변수와 파일 참조 변수는 적절히 매개변수 표시문자에 대한 값의 소스로 제공될 수 있습니다.

DESCRIPTOR *descriptor-name*

호스트 변수의 유효한 설명이 포함되어 있는 SQLDA를 식별합니다.

OPEN문이 처리되기 전에, 사용자는 다음 필드를 SQLDA에 설정해야 합니다.

- SQLDA에 제공되는 SQLVAR 어커런스 수를 나타내기 위한 SQLN.
- SQLDA에 대해 할당되는 저장영역의 바이트 수를 나타내기 위한 SQLDABC.
- 명령문 처리시 SQLDA에서 사용되는 변수의 수를 나타내기 위한 SQLDA.
- 변수의 속성을 나타내기 위한 SQLVAR 어커런스 수.

SQLDA는 모든 SQLVAR 어커런스가 들어가기에 충분한 저장영역이 있어야 합니다. 따라서 SQLDABC의 값은 16 + SQLN*(N)보다 크거나 같아야 합니다. 여기서 N은 SQLVAR 발생의 길이입니다.

LOB 결과 컬럼을 조정할 필요가 있는 경우, 모든 선택 목록 항목(또는 결과 테이블 컬럼)에 대해 두 개의 SQLVAR 항목이 있게 됩니다. 1274 페이지의 『SQLDA에서 DESCRIBE의 영향』에서 참조하십시오. SQLDOUBLED와 LOB 컬럼에 대해 설명되어 있습니다.

SQLD는 0 이상이거나 SQLN 이하의 값으로 설정되어야 합니다. 1267 페이지의 『부록C. SQL 설명자 영역(SQLDA)』에서 자세한 정보를 참조하십시오.

규칙

- 커서의 SELECT문이 평가될 때, 명령문의 각 매개변수 표시문자는 해당 호스트 변수로 대체됩니다. 입력된 매개변수 표시문자의 경우, 목표 변수 속성은 CAST 명세로 지정된 속성입니다. 미입력 매개변수 표시문자의 경우, 목표 변수 속성은 매개변수 표시문자의 문맥에 따라 결정됩니다. 매개변수 표시문자에 영향을 주는 규칙에 관해서는 1088 페이지의 『규칙』에서 참조하십시오.
- 매개변수 표시문자 P에 해당하는 호스트 변수로 V를 지정하도록 하십시오. V 값은 컬럼에 값을 지정하는 규칙에 따라 P에 대한 목표 변수에 할당됩니다. 따라서,
  - V는 목표와 호환성이 있어야 합니다.
  - V가 문자열인 경우, 그 길이는 목표 속성 길이보다 작아야 합니다.
  - V가 숫자인 경우, 정수 부분의 절대값은 목표의 정수 부분의 최대 절대값 길이보다 작아야 합니다.
  - V 속성이 목표 속성과 같지 않은 경우, 그 값은 목표 속성에 일치하도록 변환됩니다.

  커서의 SELECT문이 평가될 때, P 대신 사용되는 값은 P에 대한 목표 변수의 값입니다. 예를 들어 V가 CHAR(6)이고 목표가 CHAR(8)인 경우, P 대신에 사용된 값은 두 개의 공백이 채워진 V의 값입니다.
- USING절은 매개변수 표시문자가 들어 있는 준비된 SELECT문을 위한 것입니다. 그러나, 커서의 SELECT문이 DECLARE CURSOR문의 일부일 때에도 사용됩니다. 이런 경우, OPEN문은 목표 변수의 속성이 SELECT문의 호스트 변수 속성과 같을 때를 제외하고, SELECT문의 각 호스트 변수가 매개변수 표시

문자인 것처럼 실행됩니다. 그 결과 커서의 SELECT문에 있는 호스트 변수 값은 USING절에 지정된 호스트변수 값으로 대체됩니다.

주

- ***커서의 닫힌 상태:*** 프로그램이 시작될 때와 프로그램이 ROLLBACK문을 시작할 때 프로그램의 모든 커서를 닫힌 상태입니다.

  WITH HOLD로 선언된 열린 커서를 제외한 모든 커서는 프로그램이 COMMIT문을 발행할 때 닫힌 상태에 있습니다.

  CLOSE문이 실행되었거나 커서 위치를 예측 불가능하게 만드는 오류가 검출되어, 커서도 닫힌 상태에 있을 수 있습니다.
- 커서의 결과 테이블에서 행을 검색하려면, 커서가 열릴 때 FETCH문을 실행하십시오. 닫힌 상태에서 열린 상태로 커서 상태를 변경하는 유일한 방법은 OPEN문을 실행하는 것입니다.
- ***임시 테이블의 효과:*** 일부 경우에는, 커서의 결과 테이블이 FETCH문 실행 동안 파생됩니다. 임시 테이블 방법이 대신 사용되는 경우도 있습니다. 이러한 방법으로 OPEN문 실행중에 전체 결과 테이블이 임시 테이블로 전송됩니다. 임시 테이블이 사용될 때 프로그램의 결과는 다음 두 가지 방법에 따라 다를 수 있습니다.
  - FETCH문이 나타날 때까지 OPEN일 때 오류가 발생할 수 있습니다.
  - 커서가 열린 동안 같은 트랜잭션에서 실행되는 INSERT, UPDATE, DELETE문은 결과 테이블에 영향을 줄 수 없습니다.

  반대로 임시 테이블이 사용되지 않으면, 커서가 열려 있는 동안 실행되는 INSERT, UPDATE, DELETE문은 같은 작업 단위(UOW)로부터 발행되는 경우, 결과 테이블에 영향을 줄 수 있습니다. *응용프로그램 개발 안내서*에는 현재의 작업 단위(UOW)에서 실행한 INSERT, UPDATE 및 DELETE 조작의 효과를 제어하기 위한 잠금의 사용 방식이 설명되어 있습니다 사용자의 결과 테이블은 사용자 자신의 작업 단위(UOW)에 의해 영향받을 수도 있고, 그러한 조작의 결과는 항상 예측 가능한 것은 아닙니다. 예를 들어, 커서 C가 SELECT * FROM T로 정의된 결과 테이블 행에 있고 새로운 행이 T에 삽입된 경우,

그 행이 순서대로 지정되어 있지 않으므로 결과 테이블에 대한 삽입 효과는 예측할 수 없습니다. 따라서 후속 FETCH C는 새로운 T행을 검색할 수도 있고 하지 않을 수도 있습니다.

- 명령문 캐쉬는 OPEN문에 의해 열린 것으로 선언된 커서에 영향을 줍니다. 1017 페이지의 『주』에서 자세한 정보를 참조하십시오.

예

예 1: COBOL 프로그램으로 다음과 같은 역할을 하는 Embedded문을 작성합니다.

1. (ADMRDEPT) 부서 'A00'에 의해 관리되는 부서에 대한 DEPARTMENT 테이블로부터 모든 행을 검색하는 데 사용될 커서 C1을 정의합니다.
2. 페치(fetch)될 첫번째 행 앞에 커서 C1을 둡니다.

```
EXEC SQL  DECLARE C1 CURSOR FOR
    SELECT DEPTNO, DEPTNAME, MGRNO
  FROM DEPARTMENT
                  WHERE ADMRDEPT = 'A00'
END-EXEC.
EXEC SQL  OPEN C1
END-EXEC.
```

예 2: 커서 DYN_CURSOR를 C 프로그램에서 동적으로 정의된 선택문과 연관시키기 위해 OPEN문을 코드화하십시오. 두 개의 매개변수 표시문자가 select문의 술어 부분에 사용되었다고 가정하면, OPEN문과 함께 제공되는 두 개의 호스트 변수가 정수와 varchar(64) 값을 응용프로그램과 데이터베이스 사이에 전달합니다. (관련 호스트 변수 정의, PREPARE문 및 DECLARE CURSOR문은 또한 아래의 예에 나타나 있습니다.)

```
EXEC SQL  BEGIN DECLARE SECTION;
  static short    hv_int;
  char            hv_vchar64[64];
  char            stmt1_str[200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;
EXEC SQL  OPEN DYN_CURSOR USING :hv_int, :hv_vchar64;
```

예 3: 예 2에서와 같이 OPEN문을 코드화하지만, 여기서는 WHERE절에 있는 매개변수 표시문자의 수와 데이터 유형이 확인되지 않았습니다.

```
EXEC SQL  BEGIN DECLARE SECTION;
  char    stmt1_str[200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;
EXEC SQL  OPEN DYN_CURSOR USING DESCRIPTOR :sqlda;
```

PREPARE

SQL문이 실행되도록 동적으로 준비하기 위해 응용프로그램에 의해 PREPARE문이 사용됩니다. PREPARE문은 *명령문 문자열*이라는 문자열 형태의 명령문에서 *준비된 명령문*이라는 실행 가능 SQL문을 작성합니다.

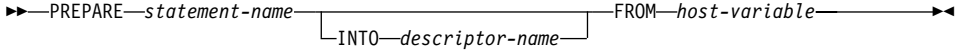
호출

이 명령문은 응용프로그램에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

명령문 준비 시간에 권한 점검이 수행되는 명령문(DML)의 경우, 명령문의 권한 ID에서 가지는 특권에는 PREPARE문에 의해 지정된 SQL문을 실행하는 데 필요한 특권이 포함되어야 합니다. 명령문 실행시 권한 부여 점검이 수행되는 명령문(DDL, GRANT 및 REVOKE)의 경우, 명령문을 사용하는 데 필요한 권한은 없으나, 준비된 명령문이 실행될 때 권한이 점검됩니다.

구문



설명

statement-name

준비된 명령문을 명명합니다. 이름이 기존의 준비된 명령문을 식별하는 경우, 그 이전에 준비된 명령문은 무효가 됩니다. 이 설명은 열린 커서의 SELECT문인 준비된 명령문을 식별해서는 안 됩니다.

INTO

INTO가 사용되고 PREPARE문이 성공적으로 실행되는 경우, 준비된 명령문에 대한 정보는 설명자 이름에 의해 지정된 SQLDA에 놓입니다.

descriptor-name

SQLDA의 이름입니다.[103]

FROM

명령문 문자열을 도입합니다. 이 명령문 문자열은 지정된 호스트 변수의 값입니다.

host-variable

문자열 변수 선언 규칙에 따라 프로그램에 기술된 호스트 변수를 식별해야 합니다. 문자열 변수(고정 길이 또는 가변 길이)이어야 합니다.

규칙

- ***명령문 문자열에 대한 규칙:*** 명령문 문자열은 동적으로 준비할 수 있는 실행 명령문이어야 합니다. 명령문 문자열은 다음 SQL문 중 하나여야 합니다.
  - ALTER
  - COMMENT ON
  - COMMIT
  - CREATE
  - DECLARE GLOBAL TEMPORARY TABLE
  - DELETE
  - DROP
  - EXPLAIN
  - FLUSH EVENT MONITOR
  - GRANT
  - INSERT
  - LOCK TABLE
  - REFRESH TABLE
  - RELEASE SAVEPOINT
  - RENAME TABLE

103. DESCRIBE문이 이 절에 대신 사용될 수도 있습니다. 974 페이지의 『DESCRIBE』에서 참조하십시오.

- RENAME TABLESPACE
- REVOKE
- ROLLBACK
- SAVEPOINT
- select-statement
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET EVENT MONITOR STATE
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION
- UPDATE

- ***매개변수 표시문자:*** 명령문 문자열에 호스트 변수 참조는 포함될 수 없으나, *매개변수 표시문자*는 포함될 수 있습니다. 명령문 문자열은 준비된 명령문 실행시 호스트 변수의 값으로 대체할 수 있습니다. 매개변수 표시문자는 물음표(?)로서 호스트 변수가 명령문 문자열이 정적 SQL문인지를 말할 수 있는 위치에서 선언됩니다. 매개변수 표시문자가 값들로 교체되는 방식에 관한 설명을 보려면, 1081 페이지의 『OPEN』과 1016 페이지의 『EXECUTE』에서 참조하십시오.

  매개변수 표시문자에는 두 가지 유형이 있습니다.

  입력된 매개변수 표시문자

  목표 데이터 유형과 함께 지정된 매개변수 표시문자. 다음과 같은 일반 형태를 지닙니다.

  ```
  CAST(? AS data-type)
  ```

이 표기는 함수 호출이 아니라, 런타임 매개변수의 유형이 지정된 데이터 유형이거나 지정된 데이터 유형으로 변환할 수 있는 데이터 유형이라는 "약속"입니다. 예를 들면, 다음과 같은 표기에서,

```
UPDATE EMPLOYEE
  SET LASTNAME = TRANSLATE(CAST(? AS VARCHAR(12)))
 WHERE EMPNO = ?
```

TRANSLATE 함수의 인수 값은 런타임 제공됩니다. 그 값의 데이터 유형은 VARCHAR(12)이거나 VARCHAR(12)로 변환될 수 있는 유형이 될 것입니다.

미입력 매개변수 표시문자

목표 데이터 유형 없이 지정되는 매개변수 표시문자. 작은 따옴표 형태를 지닙니다. 미입력 매개변수 표시문자의 데이터 유형은 문맥으로 제공됩니다. 예를 들어, 위 갱신 명령문의 술어에 있는 미입력 매개변수 표시문자는 EMPNO 컬럼의 데이터 유형과 같습니다.

입력된 매개변수 표시문자는 호스트 변수가 지원되는 동적 SQL문에서 사용될 수 있고, 그 데이터 유형은 CAST 함수에서의 약속에 기초합니다.

미입력 매개변수 표시문자는 호스트 변수가 지원되는 선택 위치에 있는 동적 SQL문에서 사용될 수 있습니다. 이러한 위치 및 그 결과 산출되는 데이터 유형은 표28에 있습니다. 그 위치들은 함께 모여 이 테이블에서 표현식, 술어 및 함수 등으로 되어, 미입력 매개변수 표시문자의 적용 가능성을 결정하는 것을 돕습니다. 미입력 매개변수 표시문자가 함수에 사용되면(산술 연산자, CONCAT 및 시간 연산자), 규정자가 함수 차수를 위해 'SYSIBM'에 설정됩니다.

표 28. 미입력 매개변수 표시문자 사용

미입력 매개변수 표시문자 위치	데이터 유형
표현식(선택 목록, CASE 및 VALUES 등)	
선택 목록에서 독자적으로	오류
연산자 우선순위 및 조작 규칙 순서를 고려한 후, 단일 산술 연산자의 두 피연산자. 다음과 같은 경우가 포함됩니다. `? + ? + 10`	오류

표 28. 미입력 매개변수 표시문자 사용 (계속)

미입력 매개변수 표시문자 위치	데이터 유형
산술식(날짜/시간 표현식이 아닌)에서 단일 연산자의 한 피연산자. 다음과 같은 경우가 포함됩니다. `? + ? * 10`	다른 피연산자의 데이터 유형
날짜/시간 표현식 내에서 레이블이 표시된 지속기간.(단위 유형을 나타내는 레이블 표시된 지속기간의 일부는 매개변수 표시문자가 될 수 없습니다.)	DECIMAL(15,0)
날짜/시간 표현식의 다른 피연산자(예: 'timecol + ?' 또는 '? - datecol').	오류
CONCAT 연산자의 두 피연산자	오류
다른 피연산자가 비CLOB 문자 데이터 유형인 CONCAT 연산자의 한 피연산자	한 피연산자가 CHAR(n) 또는 VARCHAR(n)일 경우(여기서, n은 128 미만임), 다른 피연산자는 VARCHAR(254 - n)입니다. 다른 모든 경우에 데이터 유형은 VARCHAR(254)입니다.
다른 피연산자가 비DBCLOB 그래픽 데이터 유형인 CONCAT 연산자의 한 피연산자.	한 피연산자가 GRAPHIC(n) 또는 VARGRAPHIC(n)일 경우(여기서, n은 64미만임), 다른 피연산자는 VARCHAR(127 - n)입니다. 다른 모든 경우에 데이터 유형은 VARCHAR(127)입니다.
다른 피연산자가 대형 오브젝트(LOB) 문자열인 CONCAT 연산자의 피연산자.	다른 피연산자의 데이터 유형과 같음.
UPDATE문의 SET절 오른쪽에 있는 값.	컬럼의 데이터 유형. 컬럼이 사용자 정의 구별 유형으로 정의되어 있는 경우, 그것은 사용자 정의 구별 유형의 원시 데이터 유형입니다. 컬럼이 사용자 정의 구조화 유형으로 정의된 경우, 이 컬럼은 구조화 유형이며 이는 변환 함수의 리턴 유형을 나타냅니다.
단순한 CASE 표현식에서 CASE 다음에 오는 표현식	오류
나머지 결과 표현식이 미입력 매개변수 표시문자 또는 널(NULL)인 CASE 표현식(Simple 및 Searched)에 있는 결과 표현식 중 적어도 하나.	오류

표 28. 미입력 매개변수 표시문자 사용 (계속)

미입력 매개변수 표시문자 위치	데이터 유형
단순한 CASE 표현식에서 WHEN 다음에 오는 모든 표현식.	125 페이지의 『결과 데이터 유형 규칙』을 CASE 다음에 오는 표현식과 미입력 매개변수 표시문자가 아닌 WHEN 다음에 오는 표현식에 적용시킨 결과.
적어도 하나의 결과 표현식은 NULL이 아니고 미입력 매개변수 표시문자가 아닌 CASE 표현식(Simple 및 Searched)에서의 결과 표현식.	결과 데이터 유형 규칙을 널(NULL) 또는 미입력 매개변수 표시문자가 아닌 모든 결과 표현식에 적용시킨 결과.
INSERT문 내에 없는 단일 행의 VALUES절에 있는 단일 컬럼 표현식.	오류
INSERT문 내에 없는 복수행의 VALUES절에 있는 단일 컬럼 표현식으로, 다른 모든 행 표현식에서 동일한 위치에 있는 컬럼 표현식이 미입력 매개변수 표시문자입니다.	오류
INSERT문내에 있지 않는 복수행의 VALUES절에 있는 단일 컬럼 표현식으로, 이에 대해서는 적어도 다른 하나의 행 표현식에서 동일한 위치에 있는 표현식이 미입력 매개변수 표시문자 또는 널(NULL)이 아닙니다.	125 페이지의 『결과 데이터 유형 규칙』을 미입력 매개변수 표시문자가 아닌 모든 피연산자에 적용시킨 결과.
INSERT문 내에 있는 단일 행의 VALUES절에 있는 컬럼 표현식.	컬럼의 데이터 유형. 컬럼이 사용자 정의 구별 유형으로 정의되어 있는 경우, 그것은 사용자 정의 구별 유형의 원시 데이터 유형입니다. 컬럼이 사용자 정의 구조화 유형으로 정의된 경우, 이 컬럼은 구조화 유형이며 이는 변환 함수의 리턴 유형을 나타냅니다.
INSERT문 내에 있는 복수 행의 VALUES절에 있는 컬럼 표현식.	컬럼의 데이터 유형. 컬럼이 사용자 정의 구별 유형으로 정의되어 있는 경우, 그것은 사용자 정의 구별 유형의 원시 데이터 유형입니다. 컬럼이 사용자 정의 구조화 유형으로 정의된 경우, 이 컬럼은 구조화 유형이며 이는 변환 함수의 리턴 유형을 나타냅니다.
SET 특수 레지스터 명령문의 오른쪽에 있는 값.	특수 레지스터의 데이터 유형.
술어	
비교 연산자의 두 피연산자	오류
미입력 매개변수 표시문자외의 다른 피연산자인 비교 연산자의 한 피연산자.	다른 피연산자의 데이터 유형
BETWEEN 술어의 모든 피연산자	오류

표 28. 미입력 매개변수 표시문자 사용 (계속)

미입력 매개변수 표시문자 위치	데이터 유형
다음 중 하나 BETWEEN 술어의 첫번째 및 두 번째 또는 첫번째 및 세번째 피연산자	비 매개변수 표시문자의 데이터 유형과 동일함.
나머지 BETWEEN 상황(즉, 하나의 미입력 매개변수 표시문자에 한함)	125 페이지의 『결과 데이터 유형 규칙』을 미입력 매개변수 표시문자가 아닌 모든 피연산자에 적용시킨 결과.
IN 술어의 모든 피연산자	오류
IN 술어의 첫번째와 두 번째 피연산자	미입력 매개변수 표시문자가 아닌 IN 목록의 모든 피연산자(IN 키워드 오른쪽의 피연산자)에 결과 데이터 유형 규칙을 적용시킨 결과.
오른쪽이 fullselect인 IN 술어의 첫번째 피연산자.	선택된 컬럼의 데이터 유형
IN 술어의 IN 목록에 있는 모든 피연산자	미입력 매개변수 표시문자가 아닌 IN 술어의 모든 피연산자(IN 술어의 왼쪽과 오른쪽에 있는 피연산자)에 대해 결과 데이터 유형 규칙을 적용한 결과.
IN 목록의 첫번째 피연산자를 제외한 IN 목록 내에 있는 제로 또는 그 이상의 피연산자 및 첫번째 피연산자.	미입력 매개변수 표시문자가 아닌 IN 목록의 모든 피연산자(IN 키워드 오른쪽의 피연산자)에 결과 데이터 유형 규칙을 적용시킨 결과.
LIKE 술어의 세 피연산자 모두	일치 표현식(피연산자 1)과 유형 표현식(피연산자 2)은 VARCHAR(32672)입니다. 이탈 표현식(피연산자 3)은 VARCHAR(2)입니다.
유형 표현식 또는 이탈 표현식이 미입력 매개변수 표시문자가 아닌 경우 LIKE 술어의 일치 표현식.	미입력 매개변수 표시문자가 아닌 첫번째 피연산자의 데이터 유형에 따라 VARCHAR(32672) 또는 VARGRAPHIC(16336).
일치 표현식 또는 이탈 표현식이 미입력 매개변수 표시문자가 아닌 경우 LIKE 술어의 패턴 표현식.	미입력 매개변수 표시문자가 아닌 첫번째 피연산자의 데이터 유형에 따라 VARCHAR(32672) 또는 VARGRAPHIC(16336). 일치 표현식의 데이터 유형이 BLOB이면, 패턴 표현식의 데이터 유형이 BLOB(32672)로 가정됩니다.
일치 표현식 또는 이탈 표현식이 미입력 매개변수 표시문자가 아닌 경우 LIKE 술어의 이탈 표현식.	미입력 매개변수 표시문자가 아닌 첫번째 피연산자의 데이터 유형에 따라 VARCHAR(2) 또는 VARGRAPHIC(1). 일치 표현식 또는 유형 표현식의 데이터 유형이 BLOB인 경우, 일치 표현식의 데이터 유형은 BLOB(1)인 것으로 간주됩니다.

표 28. 미입력 매개변수 표시문자 사용 (계속)

미입력 매개변수 표시문자 위치	데이터 유형
널(NULL) 술어의 피연산자	오류
	함수
COALESCE(VALUE라고도 함) 또는 NULLIF의 모든 피연산자	오류
최소한 하나의 피연산자가 미입력 매개변수 표시문자가 아닌 COALESCE의 피연산자.	125 페이지의 『결과 데이터 유형 규칙』을 미입력 매개변수 표시문자가 아닌 모든 피연산자에 적용시킨 결과.
유형화되지 않은 매개변수 마커가 아닌 다른 연산자를 가진 NULLIF 연산자.	다른 피연산자의 데이터 유형
POSSTR(두 피연산자)	두 피연산자가 모두 VARCHAR(32672)입니다.
POSSTR(다른 피연산자가 문자 데이터 유형인 한 피연산자).	VARCHAR(32672).
POSSTR(다른 피연산자가 그래픽 데이터 유형인 한 피연산자.	VARGRAPHIC(16336).
POSSTR(다른 피연산자가 BLOB인 검색 문자열 피연산자).	BLOB(32672).
SUBSTR(첫번재 피연산자)	VARCHAR(32672)
SUBSTR(두 번째와 세 번째 피연산자)	INTEGER
TRANSLATE 스칼라 함수의 첫번째 피연산자.	오류
TRANSLATE 스칼라 함수의 두 번째와 세 번째 피연산자.	첫번째 피연산자가 문자 유형일 경우 VARCHAR(32672). 첫번째 피연산자가 그랙픽 유형일 경우 VARGRAPHIC(16336).
TRANSLATE 스칼라 함수의 네 번째 피연산자.	첫번째 피연산자가 문자 유형일 경우 VARCHAR(1). 첫번째 피연산자가 그래픽 유형인 경우 VARGRAPHIC(1).
TIMESTAMP 스칼라 함수의 두 번째 피연산자.	TIME
단항 빼기	DOUBLE PRECISION
단항 더하기	DOUBLE PRECISION
사용자 정의 함수를 포함하여 다른 모든 스칼라 함수의 다른 모든 피연산자.	오류
컬럼 함수의 피연산자	오류

주

- PREPARE문이 실행될 때 명령문 문자열이 분석되고 오류가 검사됩니다. 명령문 문자열이 유효하지 않다면 오류 조건이 SQLCA에 보고됩니다. 오류가 정정되지 않는 한, 이 명령문을 참조하는 모든 후속 EXECUTE 또는 OPEN문 또한 동일한 오류를 수신합니다(시스템에서 수행된 내재적인 준비로 인해).
- 준비된 명령문은 다음과 같은 제약점을 가지고 다음 종류의 명령문에서 참조될 수 있습니다.

In...	**준비된 명령문...**
DECLARE CURSOR	SELECT여야 함
EXECUTE	SELECT여서는 *안됨*

- 준비된 명령문은 여러 번 실행될 수 있습니다. 준비된 명령문이 한 번 이상 실행되지 않고 매개변수 표시문자가 포함되지 않는 경우, PREPARE문과 EXECUTE문을 사용하지 않고 EXECUTE IMMEDIATE문을 사용하는 것이 더 효과적입니다.
- 명령문 캐싱은 반복되는 준비에 영향을 미칩니다. 1017 페이지의 『주』에서 자세한 정보를 참조하십시오.
- 지원되는 호스트 언어로 된 동적 SQL문의 예를 보려면 *응용프로그램 개발 안내서*에서 참조하십시오.

예

예 1: COBOL 프로그램에서 비 select문을 준비하여 실행합니다. 명령문이 호스트 변수 HOLDER에 포함되어 있고, 사용자의 지시에 따라 프로그램이 명령문 문자열을 그 호스트 변수에 위치시킨다고 가정합니다. 준비될 명령문은 매개변수 표시문자를 가지지 않습니다.

```
EXEC SQL  PREPARE STMT_NAME FROM :HOLDER
END-EXEC.
EXEC SQL  EXECUTE STMT_NAME
END-EXEC.
```

예 2: 예 1에서와 같이 비 select문을 준비하여 실행하지만, C 프로그램용으로 코드합니다. 준비될 명령문에는 매개변수 표시문자가 얼마든지 들어갈 수 있다고 가정합니다.

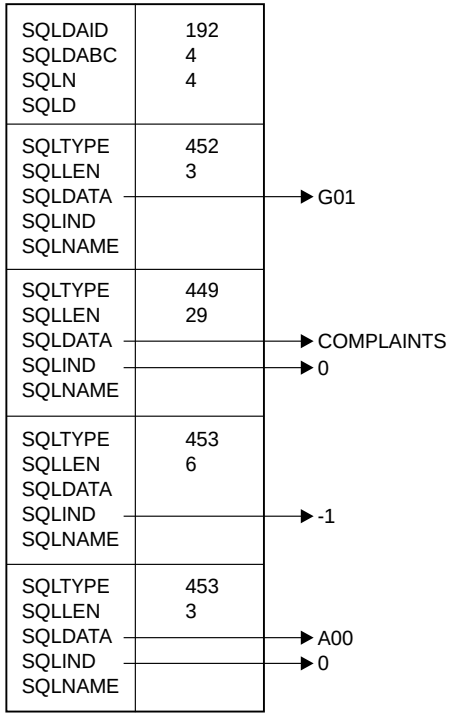
```
EXEC SQL  PREPARE STMT_NAME FROM :holder;
EXEC SQL  EXECUTE STMT_NAME USING DESCRIPTOR :insert_da;
```

다음 명령문이 준비될 것이라고 가정합니다.

```
INSERT INTO DEPT VALUES(?, ?, ?, ?)
```

DEPT 테이블의 컬럼이 다음과 같이 정의되어 있습니다.

```
DEPT_NO   CHAR(3) NOT NULL, -- 부서 번호
DEPTNAME  VARCHAR(29), -- 부서 이름
MGRNO     CHAR(6), -- 관리자 번호
ADMRDEPT  CHAR(3)  -- 행정부서 번호
```



관리자가 없고 부서 AOO에 보고하며 부서 번호가 G01인 COMPLAINTS를 삽입하기 위해서는 EXECUTE문을 발행하기 전에 구조 INSERT_DA가 위의 값을 가져야 합니다.

REFRESH TABLE

REFRESH TABLE문은 요약 테이블의 데이터를 화면갱신합니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- SYSADM 또는 DBADM 권한
- 테이블에 대한 CONTROL 특권

구문

설명

table-name

테이블 이름을 지정합니다.

내재적 또는 명시적 스키마를 포함한 이름은 현재 서버에 이미 존재하는 테이블을 식별해야 합니다. 테이블에서는 REFRESH TABLE문을 사용할 수 있어야 합니다(SQLSTATE 42809). 여기에는 다음으로 정의된 요약 테이블이 포함됩니다.

- REFRESH IMMEDIATE
- REFRESH DEFERRED

RELEASE(연결)

이 명령문은 하나 이상의 연결을 릴리스 보류 상태에 둡니다.

호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

필요사항 없음.

구문

주:

1 CURRENT 또는 ALL 응용프로그램 서버(AS)는 호스트 변수로만 식별됨에 유의하십시오.

설명

server-name 또는 *host-variable*

*server-name*이 들어 있는 *host-variable* 또는 지정된 *server-name*으로 응용프로그램 서버(AS)를 식별합니다.

*host-variable*가 지정되면, 길이가 8자 이하인 문자열 변수이어야 하며 표시기 변수를 포함해서는 안 됩니다. *host-variable* 내에 들어 있는 *server-name*은 왼쪽으로 정렬되어야 하고 따옴표로 구분해서는 안 됩니다.

*server-name*은 응용프로그램 서버(AS)를 식별하는 데이터베이스 별명임에 유의하십시오. 응용프로그램 리퀘스터의 지역 디렉토리에 나열되어야 합니다.

지정된 데이터베이스 별명 또는 호스트 변수에 들어 있는 데이터베이스 별명은 응용프로그램 프로세스의 기존 연결을 식별해야 합니다. 데이터베이스 별명이 기존의 연결을 식별하지 않은 경우, 오류(SQLSTATE 08003)가 발생합니다.

CURRENT

응용프로그램 프로세스의 현재 연결을 식별합니다. 응용프로그램 프로세스는 연결된 상태에 있어야 합니다. 그 외에는 오류(SQLSTATE 08003)가 발생합니다.

ALL

응용프로그램 프로세스의 모든 기존 연결을 식별합니다. 이러한 형태의 RELEASE문은 기존의 모든 응용프로그램 프로세스 연결을 해제 보류 상태로 둡니다. 따라서, 모든 연결은 다음 확약 조작시에 해제될 것입니다. 명령문이 실행될 때 아무 것도 연결되어 있지 않으면 오류나 경고가 나타나지 않습니다. 선택적 키워드 SQL이 DB2/MVS SQL 구문에 일치하도록 포함됩니다.

주

예

예 1: 응용프로그램에는 더 이상 IBMSTHDB로의 SQL 연결이 필요하지 않습니다. 다음 명령문은 다음 확약 조작중에 연결이 해제되도록 합니다.

```
EXEC SQL RELEASE IBMSTHDB;
```

예 2: 응용프로그램에는 더 이상 현재의 연결이 필요하지 않습니다. 다음 명령문은 다음 확약 조작중에 연결이 해제되도록 합니다.

```
EXEC SQL RELEASE CURRENT;
```

예 3: 응용프로그램이 확약 후 데이터베이스에 액세스할 필요는 없지만 당분간 수행을 계속해야 하는 경우, 이러한 연결을 종료하지 않는 것이 좋습니다. 다음 명령문은 모든 연결이 확약시 해제됨을 확인하기 위해 확약을 수행하기 전에 실행될 수 있습니다.

```
EXEC SQL RELEASE ALL;
```

RELEASE SAVEPOINT

RELEASE SAVEPOINT문은 응용프로그램이 더 이상 명명된 세이브포인트의 유지보수를 원치 않음을 나타내는 데 사용됩니다. 이 명령문이 호출된 후에는 세이브포인트로의 구간 복원이 더 이상 불가능합니다.

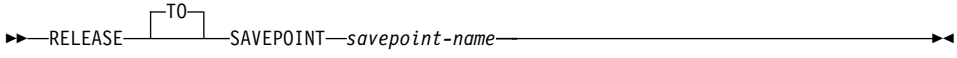
호출

이 명령문은 응용프로그램에 넣거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

필요사항 없음.

구문



설명

savepoint-name

명명된 세이브포인트가 해제됩니다. 해당 세이브포인트로의 구간 복원은 더 이상 불가능합니다. 명명된 세이브포인트가 없는 경우에는 오류가 발행됩니다 (SQLSTATE 3B001).

주

- 같은 세이브포인트 이름을 지정하는 이전의 SAVEPOINT문에 UNIQUE 키워드가 지정되었는지 여부에 관계없이 해제된 세이브포인트의 이름을 다른 SAVEPOINT문에서 다시 발행할 수 있습니다.

예

예 1: 세이브포인트 SAVEPOINT1을 해제하십시오.

```
RELEASE SAVEPOINT SAVEPOINT1
```

RENAME TABLE

RENAME TABLE문은 기존 테이블의 이름을 재명명합니다.

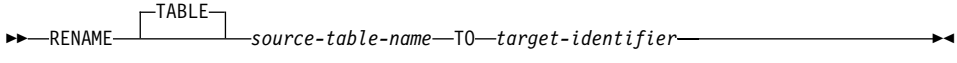
호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID가 보유한 특권에는 SYSADM 또는 DBADM 권한이나 CONTROL 특권이 포함되어야 합니다.

구문



설명

source-table-name

재명명될 기존 테이블을 지정합니다. 스키마 이름을 포함한 이름은 데이터베이스에 이미 존재하는 테이블을 식별해야 합니다(SQLSTATE 42704). 테이블을 식별하는 별명일 수도 있습니다. 이름은 카탈로그 테이블 (SQLSTATE 42832), 요약 테이블 (SQLSTATE 42997), 유형화 테이블 (SQLSTATE 42995), 별명 또는 테이블이나 별명이 아닌 오브젝트(SQLSTATE 42809)의 이름일 수 없습니다.

target-identifier

스키마 이름이 없는 테이블에 대한 새로운 이름을 지정합니다. *source-table-name*의 스키마 이름은 테이블에 대한 새로운 이름을 규정하는 데 사용됩니다. 규정화된 이름은 데이터베이스에 이미 존재하는 테이블, 뷰 또는 별명을 식별하면 안 *됩니다*(SQLSTATE 42710).

규칙

소스 테이블은

- 기존의 뷰 정의나 요약 테이블 정의내에서 참조될 수 없습니다.
- 기존 트리거의 트리거 SQL문에서 참조될 수 없거나 기존 트리거의 주제 테이블일 수 없습니다.
- SQL 함수에서 참조될 수 없습니다.
- 점검 제한조건을 가질 수 없습니다.
- 식별 컬럼이 아닌 다른 생성된 컬럼을 가질 수 없습니다.
- 참조 무결성 제한조건에서 상위 또는 종속 테이블일 수 없습니다.
- 기존의 참조된 컬럼 범위일 수 없습니다.

소스 테이블이 위 조건을 하나 이상 위반할 경우 오류가 리턴됩니다(SQLSTATE 42986).

주

- 새로운 테이블 이름을 반영하도록 카탈로그 항목이 갱신됩니다.
- 소스 테이블 이름과 연관된 모든 권한이 새로운 테이블 이름으로 *전송*됩니다.(권한 카탈로그 테이블도 그에 맞게 갱신됩니다.)
- 소스 테이블에 대해 정의된 색인이 새로운 테이블로 *전송*됩니다.(색인 카탈로그 테이블도 그에 맞게 갱신됩니다.)
- 소스 테이블에 종속되는 모든 패키지는 무효화됩니다.
- 별명이 *소스 테이블 이름*에 사용될 경우, 이것은 테이블 이름으로 환원되어야 합니다. 테이블은 이 테이블 스키마 내에서 재명명됩니다. 별명은 RENAME문에 의해 변경되지 않으며 기존의 테이블 이름을 그대로 지칭합니다.
- 외부 키로 기본 키나 고유 제한조건이 참조되지 않을 경우, 기본 키나 고유 제한조건을 갖는 테이블을 재명명할 수 있습니다.

예

EMP 테이블의 이름을 EMPLOYEE로 변경합니다.

```
RENAME TABLE EMP TO EMPLOYEE
```

RENAME TABLESPACE

RENAME TABLESPACE문은 기존 테이블 공간 이름을 재명명합니다.

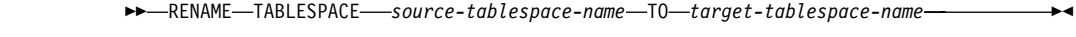
호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID가 보유한 특권에는 SYSADM 또는 SYSCTRL 권한이 포함되어야 합니다.

구문



설명

source-tablespace-name

한 부분 이름으로 이름이 변경된 기존의 테이블 공간을 지정합니다. 이것은 SQL 식별자(일반 또는 분리 식별자)입니다. 테이블 공간 이름은 카탈로그에 이미 있는 테이블 공간을 식별해야 합니다(SQLSTATE 42704).

target-tablespace-name

테이블 공간의 새 이름을 한 부분 이름으로 지정합니다. 이것은 SQL 식별자(일반 또는 분리 식별자)입니다. 새 테이블 공간 이름은 카탈로그에 이미 있는 테이블 공간을 식별*하지 않아야* 합니다(SQLSTATE 42710). 이것은 'SYS'로 시작할 수 없습니다(SQLSTATE 42939).

규칙

- SYSCATSPACE 테이블 공간은 이름을 바꿀 수 없습니다(SQLSTATE 42832).

- ″롤 포워드 보류″ 또는 ″롤 포워드 진행″ 상태인 모든 테이블 공간은 이름을 바꿀 수 없습니다(SQLSTATE 55039).

주

- 테이블 공간의 이름을 바꾸면 테이블 공간의 최소 복구 시간이 이름 바꾸기가 이루어진 시점으로 갱신됩니다. 이는 테이블 공간 레벨에서의 롤 포워드가 최소한 이 특정 시점이어야 함을 나타냅니다.
- 백업 작성후 이름 바꾸기가 수행된 백업 이미지로부터 테이블 공간을 복원할 경우에는 새 테이블 공간 이름을 사용해야 합니다. 백업 복원에 대한 정보는 *Administrative API Reference* 또는 *Command Reference*에서 참조하십시오.

예

USERSPACE1 테이블 공간의 이름을 DATA2000으로 변경합니다.

```
RENAME TABLESPACE USERSPACE1 TO DATA2000
```

REVOKE(데이터베이스 권한)

이러한 형태의 REVOKE문은 전체 데이터베이스에 적용되는 권한을 권한 취소합니다.

호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

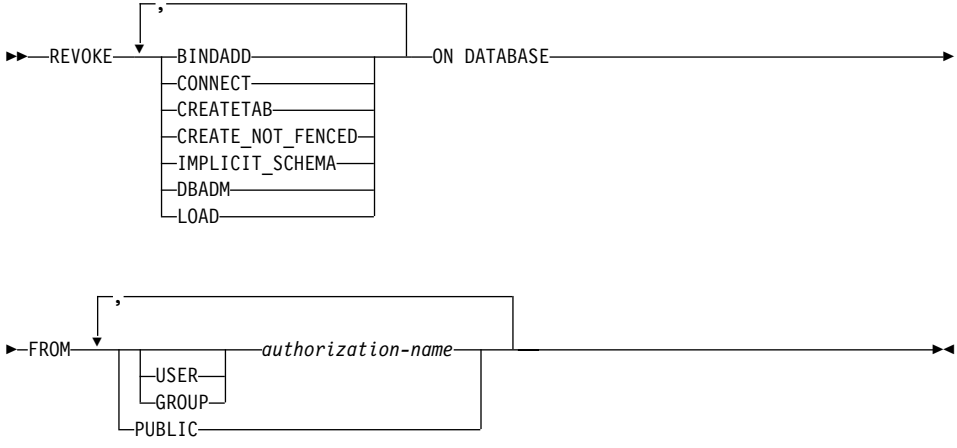
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- DBADM 권한
- SYSADM 권한

DBADM 권한을 취소하려면 SYSADM 권한이 필요합니다.

구문



설명

BINDADD

패키지 작성 권한을 취소합니다. 패키지 작성자는 자동으로 그 패키지에 대한 CONTROL 특권을 가지며, 나중에 BINDADD 권한이 취소되어도 이 특권을 보유합니다.

BINDADD 권한은 DBADM 권한도 취소하지 않고 DBADM 권한을 보유하는 *authorization-name*으로부터 취소될 수 없습니다.

CONNECT

데이터베이스에 액세스할 권한을 취소합니다.

사용자로부터 CONNECT 권한을 취소해도 데이터베이스의 오브젝트에 있는 사용자에게 권한 부여된 특권에는 영향을 미치지 않습니다. 사용자가 나중에 다시 CONNECT 권한을 받으면 이전에 보유했던 모든 특권이 여전히 유효하게 됩니다.(즉, 권한이 명시적으로 권한 취소되지 않은 것으로 간주됩니다.)

CONNECT 권한은 DBADM 권한도 취소하지 않고 DBADM 권한을 보유하는 *authorization-name*으로부터 취소될 수 없습니다.

CREATETAB

테이블 작성 권한을 취소합니다. 테이블 작성자는 자동으로 그 테이블에 대한 CONTROL 특권을 갖게 되며, 이 특권은 나중에 작성자의 CREATETAB 권한이 취소되어도 그대로 남아 있습니다.

CREATETAB 권한은 DBADM 권한도 취소하지 않고 DBADM 권한을 보유하는 *authorization-name*으로부터 취소될 수 없습니다(SQLSTATE 42504).

CREATE_NOT_FENCED

데이터베이스 관리 프로그램 프로세스에서 실행되는 함수를 등록하는 권한을 취소합니다. 그러나, 일단 함수가 보호되지 않은 것으로 등록되면, 나중에 이 함수를 등록한 권한 부여 ID로부터 CREATE_NOT_FENCED가 권한 취소되더라도 이러한 방식으로 계속 수행됩니다.

CREATE_NOT_FENCED 권한은 DBADM 권한도 취소하지 않고 DBAM 권한을 보유하는 *authorization-name*으로부터 권한 취소될 수 없습니다 (SQLSTATE 42504).

IMPLICIT_SCHEMA

스키마를 내재적으로 작성하기 위한 권한을 취소합니다. 이것은 기존 스키마에서 오브젝트를 작성하거나 CREATE SCHEMA문을 프로세스하는 데에는 영향을 주지 않습니다.

DBADM

DBADM 권한을 취소합니다.

DBADM 권한은 (PUBLIC으로 권한 부여될 수 없으므로) PUBLIC으로부터 취소될 수 없습니다.

DBADM 권한을 취소하면 데이터베이스내의 오브젝트에 대해 권한 부여 이름으로 보류된 특권이 자동 취소되지 않거나 BINDADD, CONNECT, CREATETAB, IMPLICIT_SCHEMA 또는 CREATE_NOT_FENCED 권한이 취소되지 않습니다.

LOAD

이 데이터베이스에서 로드 권한을 취소합니다.

FROM

권한이 취소된 출처를 표시합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

하나 이상의 권한 부여 ID를 나열합니다.

REVOKE문의 권한 부여 ID는 그 자체로는 사용될 수 없습니다 (SQLSTATE 42502). REVOKE문의 권한 부여 이름과 같은 *authorization-name*에서 특권을 취소하는 것은 불가능합니다.

PUBLIC

PUBLIC에서 권한을 취소합니다.

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,

- SYSCAT.DBAUTH 카탈로그 뷰(view)에 권한 받은 사용자에 대한 모든 행에 U의 GRANTEETYPE이 있는 경우, USER로 간주합니다.
- 모든 행에 G의 GRANTEETYPE이 있는 경우, GROUP으로 간주합니다.
- 어떤 행에 U가 있고 어떤 행에 G가 있는 경우, 오류(SQLSTATE 56092)가 발생합니다.
- DCE 인증이 사용될 경우, 오류가 발생합니다(SQLSTATE 56092).

주

- 특정의 특권을 권한 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. PUBLIC 또는 그룹이 다른 특권을 부여하거나 DBADM과 같은 상위 레벨 권한을 가질 경우, 사용자는 이들 타스크를 처리할 수 있습니다.

예

예 1: USER6이 유일한 사용자이고 그룹이 아닌 상태에서 사용자 USER6으로부터 테이블을 작성하는 특권을 권한 취소합니다.

```
REVOKE CREATETAB ON DATABASE FROM USER6
```

예 2: 그룹 D024에서 데이터베이스에 대한 BINDADD 권한을 취소합니다. SYSCAT.DBAUTH 카탈로그 뷰에는 권한 받은 사용자에 대한 두 개의 행이 있습니다. 하나는 U의 GRANTEETYPE이고 다른 하나는 G의 GRANTEETYPE입니다.

```
REVOKE BINDADD ON DATABASE FROM GROUP D024
```

이런 경우, GROUP 키워드를 지정해야 합니다. 그렇지 않으면, 오류(SQLSTATE 56092)가 발생합니다.

REVOKE(색인 특권)

이러한 형태의 REVOKE문은 색인에 대한 CONTROL 특권을 권한 취소합니다.

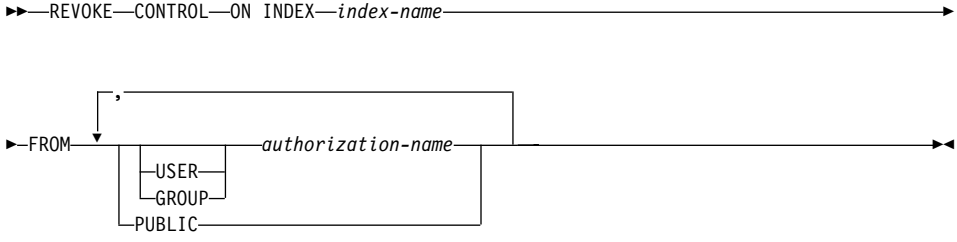
호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID는 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다(SQLSTATE 42501).

구문



설명

CONTROL

색인 삭제 특권을 권한 취소합니다. 이는 색인에 대한 CONTROL 특권으로서, 색인 작성자에게 자동으로 권한 부여됩니다.

ON INDEX *index-name*

CONTROL 특권이 권한 취소될 색인 이름을 지정합니다.

FROM

특권이 권한 취소되는 출처를 나타냅니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

하나 이상의 권한 부여 ID를 나열합니다.

REVOKE문의 권한 부여 ID는 그 자체로는 사용될 수 없습니다 (SQLSTATE 42502). REVOKE문의 권한 부여 ID와 같은 *authorization-name*에서 특권을 취소하는 것은 불가능합니다.

PUBLIC

PUBLIC에서 특권을 권한 취소합니다.

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  SYSCAT.INDEXAUTH 카탈로그 뷰의 권한 받은 사용자에 대한 모든 행에 U의 GRANTEETYPE이 있는 경우, USER로 간주합니다.
  모든 행에 G의 GRANTEETYPE이 있는 경우, GROUP으로 간주합니다.
  어떤 행에 U가 있고 어떤 행에 G가 있는 경우, 오류(SQLSTATE 56092)가 발생합니다.
  DCE 인증이 사용될 경우, 오류가 발생합니다(SQLSTATE 56092).

주

- 특정의 특권을 권한 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. PUBLIC 또는 그룹이 다른 특권을 부여하거나 색인의 스키마에 대한 ALTERIN과 같은 권한을 가질 경우, 사용자는 이들 타스크를 처리할 수 있습니다.

예

예 1: USER4가 유일한 사용자이고 그룹이 아닌 상태에서 사용자 USER4에서 색인 DEPTIDX를 삭제하기 위한 특권을 권한 취소합니다.

```
REVOKE CONTROL ON INDEX DEPTIDX FROM USER4
```

예 2: 사용자 CHEF와 그룹 WAITERS에서 색인 LUNCHITEMS를 삭제하기 위한 권한을 취소합니다.

```
REVOKE CONTROL ON INDEX LUNCHITEMS
 FROM USER CHEF, GROUP WAITERS
```

REVOKE(패키지 특권)

이러한 형태의 REVOKE문은 패키지에 대한 CONTROL, BIND, EXECUTE 특권을 권한 취소합니다.

호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

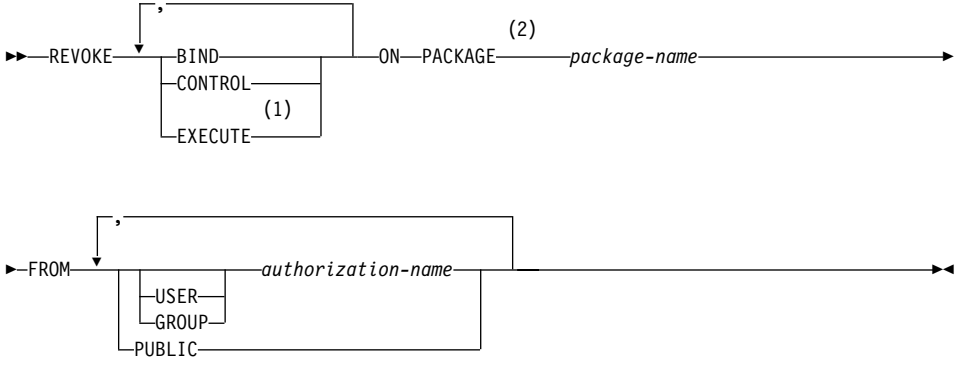
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- 참조된 패키지에 대한 CONTROL 특권
- SYSADM 또는 DBADM 권한

CONTROL 특권을 권한 취소하려면 SYSADM이나 DBADM 권한이 필요합니다.

구문



주:

1 RUN은 EXECUTE에 대한 동의어로 사용할 수 있습니다.

2 PACKAGE에 대한 동의어로 PROGRAM을 사용할 수 있습니다.

설명

BIND

언급된 패키지에서 BIND 또는 REBIND를 실행할 수 있는 특권을 권한 취소합니다.

BIND 특권은 CONTROL 특권을 권한 취소하지 않고 패키지상에 CONTROL 특권을 보유하는 *authorization-name*으로부터 취소될 수 없습니다.

CONTROL

패키지 삭제 특권을 권한 취소하고 패키지 특권을 다른 사용자에게 부여합니다.

CONTROL을 취소해도 다른 패키지 특권은 권한 취소되지 않습니다.

EXECUTE

패키지 실행 특권을 권한 취소합니다.

EXECUTE 특권은 CONTROL 특권을 권한 취소하지 않고 패키지상에 CONTROL 특권을 보유하는 *authorization-name*으로부터 취소될 수 없습니다.

ON PACKAGE *package-name*

특권이 권한 취소될 패키지를 지정합니다.

FROM

특권이 권한 취소되는 출처를 나타냅니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

하나 이상의 권한 부여 ID를 나열합니다.

REVOKE문의 권한 부여 ID는 그 자체로는 사용될 수 없습니다 (SQLSTATE 42502). REVOKE문의 권한 부여 ID와 같은 *authorization-name*에서 특권을 취소하는 것은 불가능합니다.

PUBLIC

PUBLIC에서 특권을 권한 취소합니다.

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  - SYSCAT.PACKAGEAUTH 카탈로그 뷰의 권한 받은 사용자에 대한 모든 행에 U의 GRANTEETYPE이 있는 경우, USER로 간주합니다.
  - 모든 행에 G의 GRANTEETYPE이 있는 경우, GROUP으로 간주합니다.
  - 어떤 행에 U가 있고 어떤 행에 G가 있는 경우, 오류(SQLSTATE 56092)가 발생합니다.
  - DCE 인증이 사용될 경우, 오류가 발생합니다(SQLSTATE 56092).

주

- 특정의 특권을 권한 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. PUBLIC 또는 그룹이 다른 특권을 부여하거나 패키지의 스키마에 대한 ALTERIN 같은 권한을 가질 경우, 사용자는 이들 타스크를 처리할 수 있습니다.

예

예 1: 패키지 CORPDATA.PKGA에 대한 EXECUTE 특권을 PUBLIC에서 권한 취소합니다.

```
REVOKE EXECUTE
  ON PACKAGE CORPDATA.PKGA
  FROM PUBLIC
```

예 2: 사용자 FRANK와 PUBLIC에 대해 RRSP_PKG 패키지의 CONTROL 권한을 취소하십시오.

```
REVOKE CONTROL
  ON PACKAGE RRSP_PKG
  FROM USER FRANK, PUBLIC
```

REVOKE(스키마 특권)

이러한 형태의 REVOKE문은 스키마에 대한 특권을 권한 취소합니다.

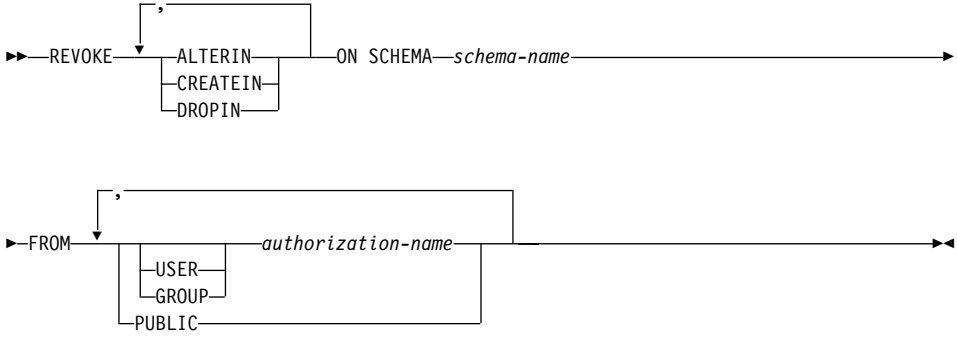
호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID는 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다(SQLSTATE 42501).

구문



설명

ALTERIN

스키마에 있는 오브젝트에 주석을 달거나 변경하기 위한 특권을 권한 취소합니다.

CREATEIN

스키마에 오브젝트를 작성하기 위한 특권을 권한 취소합니다.

DROPIN
스키마의 오브젝트를 삭제하기 위한 권한을 권한 취소합니다.

ON SCHEMA *schema-name*
특권이 권한 취소될 스키마의 이름을 지정합니다.

FROM
특권이 권한 취소되는 출처를 나타냅니다.

USER
*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP
*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...
하나 이상의 권한 부여 ID를 나열합니다.

REVOKE문의 권한 부여 ID는 그 자체로는 사용될 수 없습니다 (SQLSTATE 42502). REVOKE문의 권한 부여 ID와 같은 *authorization-name*에서 특권을 취소하는 것은 불가능합니다.

PUBLIC
PUBLIC에서 특권을 권한 취소합니다.

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  - SYSCAT.SCHEMAAUTH 카탈로그 뷰에 권한 받은 사용자에 대한 모든 행에 U의 GRANTEETYPE이 있는 경우, USER로 간주합니다.
  - 모든 행에 G의 GRANTEETYPE이 있는 경우, GROUP으로 간주합니다.
  - 어떤 행에 U가 있고 어떤 행에 G가 있는 경우, 오류(SQLSTATE 56092)가 발생합니다.
  - DCE 인증이 사용될 경우, 오류가 발생합니다(SQLSTATE 56092).

주

- 특정의 특권을 권한 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. PUBLIC 또는 그룹이 다른 특권을 부여하거나 DBADM과 같은 상위 레벨 권한을 가질 경우, 사용자는 이들 타스크를 처리할 수 있습니다.

예

예 1: USER4가 유일한 사용자이고 그룹이 아닌 상태에서 사용자 USER4에서 스키마 DEPTIDX를 작성하기 위한 특권을 권한 취소합니다.

```
REVOKE CREATEIN ON SCHEMA DEPTIDX FROM USER4
```

예 2: 사용자 CHEF와 그룹 WAITERS에서 스키마 LUNCH의 오브젝트를 삭제하기 위한 특권을 권한 취소합니다.

```
REVOKE DROPIN ON SCHEMA LUNCH
   FROM USER CHEF, GROUP WAITERS
```

REVOKE(서버 특권)

이러한 형태의 REVOKE문은 통과 모드에서 지정된 데이터 소스를 액세스하고 사용할 특권을 권한 취소합니다.

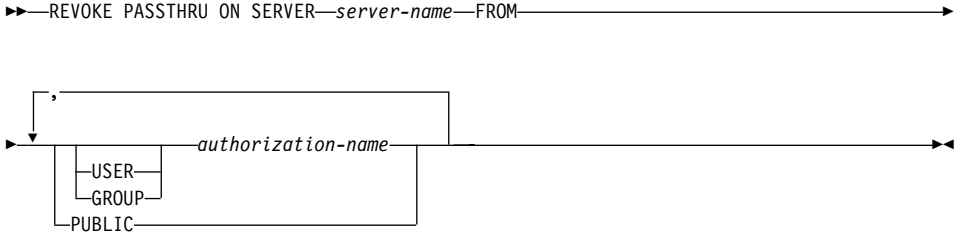
호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID는 SYSADM 또는 DBADM 권한이 있어야 합니다.

구문



설명

SERVER *server-name*

통과 모드에서 사용할 특권이 권한 취소되고 있는 데이터 소스에 이름을 부여합니다. *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다.

FROM

특권을 권한 취소한 사용자로부터 지정합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...

여러 사용자 또는 그룹의 권한 부여 ID를 나열합니다.

REVOKE문의 권한 부여 ID는 그 자체로 사용될 수는 없습니다 (SQLSTATE 42502). REVOKE문의 권한 부여 ID와 같은 *authorization-name*에서 특권을 취소하는 것은 불가능합니다.

PUBLIC

*server-name*을 통과하는 특권을 모든 사용자로부터 권한 취소합니다.

예

예 1: 데이터 소스를 통과하는 USER6 특권의 권한 취소

```
REVOKE PASSTHRU ON SERVER MOUNTAIN FROM USER USER6
```

예 2: 데이터 소스 EASTWING을 통과하는 그룹 D024 특권의 권한 취소

```
REVOKE PASSTHRU ON SERVER EASTWING FROM GROUP D024
```

그룹 D024의 구성원들은 더 이상 그들의 그룹 ID를 사용하여 EASTWING을 통과할 수 없을 것입니다. 그러나 구성원에게 그 자신의 사용자 ID하에서 EASTWING으로 통과할 수 있는 권한이 있는 경우 구성원은 이러한 특권을 유지합니다.

REVOKE(테이블, 뷰 또는 별명 특권)

REVOKE문의 이 형태는 테이블, 뷰 또는 별명에 대한 특권을 권한 취소합니다.

호출

이 명령문은 응용프로그램에 Embedded SQL문이나, 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

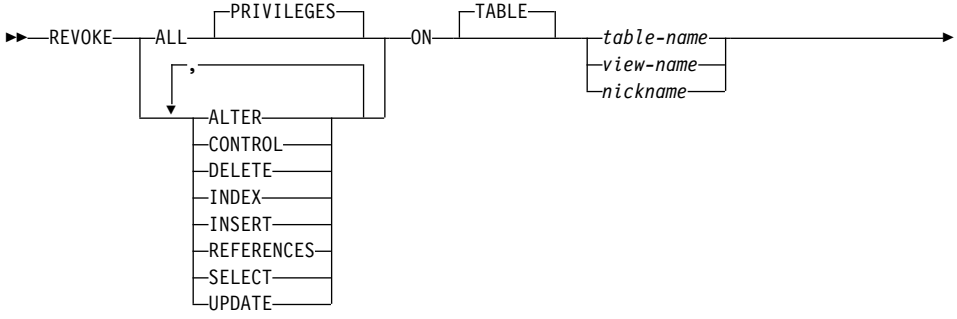
명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

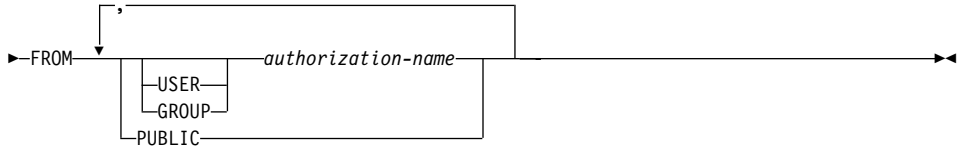
- SYSADM 또는 DBADM 권한
- 참조된 테이블, 뷰 또는 별명에 대한 CONTROL 특권.

CONTROL 특권을 권한 취소하려면 SYSADM이나 DBADM 권한이 필요합니다.

카탈로그 테이블 및 뷰에 대한 특권을 권한 취소하려면 SYSADM이나 DBADM 권한이 필요합니다.

구문





설명

ALL 또는 ALL PRIVILEGES

지정된 테이블, 뷰 또는 별명에 대한 권한 부여 이름이 보유하는 모든 특권을 권한 취소합니다.

ALL을 사용하지 않는 경우, 아래 나열된 키워드가 하나 이상 사용되어야 합니다. 각 키워드는 기술된 특권을 취소하나, ON절에 명명된 테이블, 뷰 또는 또는 별명에 적용될 때에만 권한 취소됩니다. 같은 키워드는 한 번 이상 지정될 수 없습니다.

ALTER

기본 테이블 정의에 컬럼을 추가하고, 테이블상의 기본 키나 고유성 제한조건을 작성 또는 삭제하고, 테이블, 뷰 또는 별명에 주석을 추가/변경하고, 점검 제한조건을 작성 또는 삭제하고, 트리거를 작성하고, 별명에 대한 컬럼 옵션을 추가, 재설정 또는 삭제하고, 별명 컬럼 이름이나 데이터 유형을 변경할 특권을 권한 취소합니다.

CONTROL

테이블, 뷰 또는 별명을 삭제할 수 있는 능력과 테이블 및 색인에 대해 Runstats 유틸리티를 실행할 수 있는 능력을 권한 취소합니다.

*authorization-name*으로부터 CONTROL 특권을 취소해도 그 오브젝트상의 사용자에게 권한 부여된 다른 특권은 권한 취소되지 않습니다.

DELETE

테이블 또는 갱신 가능 뷰에서 행 삭제 권한을 취소합니다.

INDEX

테이블의 색인 또는 별명의 색인 스펙 작성 특권을 권한 취소합니다. 색인이나 색인 스펙 작성자는 자동으로 색인이나 색인 스펙에 대한 CONTROL 특

권을 가집니다.(작성자에게 색인이나 색인 스펙을 삭제할 권한을 줍니다.) 또한, 작성자는 INDEX 특권이 권한 취소될 지라도 이 특권을 보유합니다.

INSERT
테이블 또는 갱신 가능 뷰로 행 삽입 특권 및 IMPORT 유틸리티 실행 특권을 권한 취소합니다.

REFERENCES
테이블을 상위 테이블로 참조하는 외부 키를 작성 또는 삭제하는 특권을 권한 취소합니다. 모든 컬럼 레벨의 REFERENCES 특권도 권한 취소됩니다.

SELECT
테이블이나 뷰에서 행을 검색하고 테이블에 뷰를 작성하며 테이블이나 뷰에 대해 EXPORT 유틸리티를 수행할 수 있는 특권을 취소합니다.

SELECT 특권을 취소하면, 일부 뷰가 실행 불능으로 표시됩니다. 실행 불능 뷰에 대해 941 페이지의 『주』에서 참조하십시오.

UPDATE
테이블이나 갱신 가능 뷰의 행을 갱신할 수 있는 특권을 취소합니다. 모든 컬럼 레벨의 UPDATE 특권도 권한 취소됩니다.

ON TABLE *table-name* 또는 *view-name* 또는 *nickname*
특권이 권한 취소될 해당 테이블, 뷰 또는 별명을 지정합니다. *table-name*은 임시 테이블로 선언될 수 없습니다(SQLSTATE 42995).

FROM
특권이 권한 취소되는 출처를 나타냅니다.

USER
*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP
*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name,...
하나 이상의 권한 부여 ID를 나열합니다.

REVOKE문의 ID는 그 자체로는 사용될 수 없습니다(SQLSTATE 42502). REVOKE문의 권한 부여 ID와 같은 *authorization-name*에서 특권을 취소하는 것은 불가능합니다.

PUBLIC

PUBLIC에서 특권을 권한 취소합니다.

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  - SYSCAT.TABAUTH 및 SYSCAT.COLAUTH 카탈로그 뷰에서 권한 받은 사용자에 대한 모든 행에 U의 GRANTEETYPE이 있는 경우, USER로 간주합니다.
  - 모든 행에 G의 GRANTEETYPE이 있는 경우, GROUP으로 간주합니다.
  - 어떤 행에 U가 있고 어떤 행에 G가 있는 경우, 오류(SQLSTATE 56092)가 발생합니다.
  - DCE 인증이 사용될 경우, 오류가 발생합니다(SQLSTATE 56092).

주

- 뷰를 작성하는 데 사용된 *권한 부여 이름*에서 특권이 권한 취소되는 경우(이것을 SYSCAT.VIEWS에 있는 뷰의 DEFINER라고 함), 그 특권은 모든 종속 뷰에서도 권한 취소됩니다.
- 뷰의 DEFINER가 뷰 정의가 종속된 일부 오브젝트에 대해 SELECT 특권을 상실한 경우(또는 뷰 정의가 종속된 오브젝트가 삭제되었거나, 다른 뷰의 경우 실행 불능 상태가 된 경우), 뷰는 실행 불능 상태가 됩니다.(실행 불능 상태의 뷰에 대한 자세한 정보는 931 페이지의 『CREATE VIEW』의 “주의” 절을 참조하십시오.)

  단, DBADM 또는 SYSADM이 명시적으로 DEFINER로부터 뷰에 대한 모든 특권을 권한 취소한 경우, DEFINER의 레코드는 SYSCAT.TABAUTH에 나타나지 않으나 뷰에는 아무 것도 발생하지 않으며 조작은 가능합니다.
- 실행 불능 뷰에 대한 특권은 권한 취소될 수 없습니다.
- 특권이 권한 취소될 오브젝트에 종속되는 모든 패키지는 유효하지 않은 것으로 표시됩니다. 응용프로그램의 바인드 또는 리바인드 조작이 정상적으로 수행되거

나, 그 응용프로그램이 실행되고 데이터베이스 관리 프로그램이 (카탈로그에 저장된 정보를 사용하여) 성공적으로 응용프로그램을 정상적으로 리바인드할 때까지 패키지는 유효하지 않은 상태로 있게 됩니다. 권한 취소로 인해 유효하지 않은 것으로 표시된 패키지는 추가로 권한을 부여할 필요없이 성공적으로 리바인드됩니다.

예를 들어, USER1이 소유한 패키지에 테이블 T1으로부터의 SELECT가 있고, 테이블 T1에 대한 SELECT 특권이 USER1에서 권한 취소되면 그 패키지는 유효하지 않은 것으로 표시됩니다. SELECT 권한이 재부여되거나, 사용자가 DBADM 권한을 가지고 있는 경우, 그 패키지는 수행시 성공적으로 리바인드됩니다.

- FROM절에서 OUTER(*Z*)의 사용을 포함하는 패키지, 트리거 또는 뷰는 *Z*의 모든 서브테이블이나 서브뷰에 SELECT 특권을 가지는 것에 종속됩니다. 마찬가지로, DEREF(*Y*)(여기서 *Y*는 목표 테이블 또는 뷰가 *Z*인 참조 유형임)의 사용을 포함하는 패키지, 트리거 또는 뷰는 *Z*의 모든 서브테이블이나 서브뷰에 SELECT 특권을 가지는 것에 종속됩니다. 이들 SELECT 특권들 중 하나가 권한 취소되면, 서브패키지가 무효화되어 트리거나 뷰가 작동 불능으로 됩니다.
- 테이블, 뷰 또는 별명 특권은 CONTROL 특권도 권한 취소하지 않고 오브젝트에 대한 CONTROL을 보유하는 *권한 부여 이름*으로부터 취소할 수 없습니다 (SQLSTATE 42504).
- 특정의 특권을 권한 취소한다고 하여 조치를 수행할 능력이 취소되는 것은 아닙니다. PUBLIC 또는 그룹이 다른 특권을 부여하거나 테이블 또는 뷰의 스키마에 대한 ALTERIN과 같은 권한을 가질 경우, 사용자는 이들 타스크를 처리할 수 있습니다.
- 요약 테이블의 DEFINER가 요약 테이블 정의가 종속된 테이블에 대한 SELECT 특권을 상실하면(또는 요약 테이블 정의가 종속된 테이블이 식제된 경우), 요약 테이블은 실행 불능 상태가 됩니다.(실행 불능 요약 테이블에 대한 849 페이지의 『주』에서 정보를 참조하십시오.)

  그러나, DBADM 또는 SYSADM이 명시적으로 DEFINER로부터 요약 테이블에 있는 모든 특권을 권한 취소하면, 그 DEFINER의 SYSTABAUTH에 있는 레코드는 삭제되나, 그 요약 테이블에는 아무 일도 발생하지 않으며 조작은 가능합니다.

- 별명 특권을 취소해도 데이터 소스 오브젝트(테이블이나 뷰) 특권에 아무 영향도 미치지 않습니다.
- 다른 몇몇 오브젝트가 SQL 함수에 종속적이기 때문에 SQL 함수를 삭제할 수 없는 경우에는 SQL 함수에서 *직접* 또는 *간접적*으로 참조되는 테이블이나 뷰에 대한 SELECT 특권 취소가 실패할 수 있습니다(SQLSTATE 42893).

주: 1003 페이지의 『규칙』에서는 테이블 및 뷰와 같은 오브젝트가 서로에 대해 가질 수 있는 종속성을 나열합니다.

예

예 1: 사용자 ENGLES에서 EMPLOYEE 테이블에 대한 SELECT 특권을 권한 취소합니다. 이 테이블 및 권한 받은 사용자에 대해 SYSCAT.TABAUTH 카탈로그 뷰에 한 행이 있고, GRANTEETYPE 값은 U입니다.

```
REVOKE SELECT
  ON TABLE EMPLOYEE
  FROM ENGLES
```

예 2: 이전에 모든 지역 사용자에게 권한 부여된 테이블 EMPLOYEE에 대한 갱신 특권을 권한 취소합니다. 특정 사용자에게 권한 부여하는 것은 영향받지 않음에 유의하십시오.

```
REVOKE UPDATE
  ON EMPLOYEE
  FROM PUBLIC
```

예 3: 사용자 PELLOW와 MLI, 그룹 PLANNERS에서 테이블 EMPLOYEE에 대한 모든 특권을 권한 취소합니다.

```
REVOKE ALL
  ON EMPLOYEE
  FROM USER PELLOW, USER MLI,
GROUP PLANNERS
```

예 4: 사용자 JOHN에게서 테이블 CORPDATA.EMPLOYEE에 대한 SELECT 특권을 권한 취소합니다. 이 테이블 및 권한 받은 사용자에 대해 SYSCAT.TABAUTH 카탈로그 뷰에 한 행이 있고, GRANTEETYPE 값은 U입니다.

```
REVOKE SELECT
  ON CORPDATA.EMPLOYEE FROM JOHN
```

또는

```
REVOKE SELECT
  ON CORPDATA.EMPLOYEE FROM USER JOHN
```

GROUP JOHN으로부터 특권을 권한 취소하려고 하면 오류가 발생함에 유의하십시오. GROUP JOHN에게 특권이 주어져 있지 않기 때문입니다.

예 5: 그룹 JOHN에게서 테이블 CORPDATA.EMPLOYEE에 대한 SELECT 특권을 권한 취소합니다. 이 테이블 및 권한 받은 사용자에 대해 SYSCAT.TABAUTH 카탈로그 뷰에 한 행이 있으며, GRANTEETYPE 값은 G입니다.

```
REVOKE SELECT
  ON CORPDATA.EMPLOYEE FROM JOHN
```

또는

```
REVOKE SELECT
 ON CORPDATA.EMPLOYEE FROM GROUP JOHN
```

예 6: 별명 ORAREM1에 대한 색인 스펙을 작성하는 사용자 SHAWN의 특권을 권한 취소하십시오.

```
REVOKE INDEX
  ON ORAREM1 FROM USER SHAWN
```

REVOKE(테이블 공간 권한 취소)

이러한 형태의 REVOKE문은 테이블에 대한 USE 특권을 권한 취소합니다.

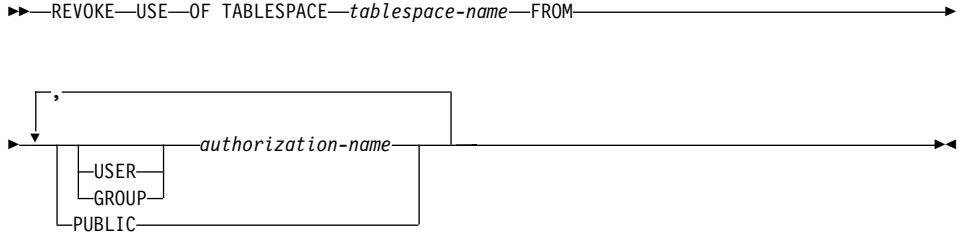
호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에는 SYSADM, SYSCTRL 또는 DBADM 권한이 있어야 합니다(SQLSTATE 42501).

구문



설명

USE

테이블 작성시 테이블 공간을 지정하거나 기본값으로 설정할 수 있는 특권을 취소합니다.

OF TABLESPACE *tablespace-name*

USE 특권이 취소될 테이블 공간을 지정합니다. 테이블 공간은 SYSCATSPACE(SQLSTATE 42838) 또는 SYSTEM TEMPORARY 테이블 공간(SQLSTATE 42809)일 수 없습니다.

FROM

USE 특권을 권한 취소한 사용자로부터 지시합니다.

USER

*authorization-name*이 사용자를 식별하도록 지정합니다.

GROUP

*authorization-name*이 그룹 이름을 식별하도록 지정합니다.

authorization-name

하나 이상의 권한 부여 ID를 나열합니다.

REVOKE 문의 권한 부여 ID는 그 자체로는 사용될 수 없습니다 (SQLSTATE 42502). REVOKE 문의 권한 부여 ID와 같은 *authorization-name*에서 특권을 취소하는 것은 불가능합니다.

PUBLIC

PUBLIC에서 USE 특권을 권한 취소합니다.

규칙

- USER와 GROUP이 모두 지정되지 않은 경우,
  - SYSCAT.PACKAGEAUTH 카탈로그 뷰의 권한 받은 사용자에 대한 모든 행에 U의 GRANTEETYPE이 있는 경우, USER로 간주합니다.
  - 모든 행에 G의 GRANTEETYPE이 있는 경우, GROUP으로 간주합니다.
  - 어떤 행에 U가 있고 어떤 행에 G가 있는 경우, 오류가 발생합니다 (SQLSTATE 56092).
  - DCE 인증이 사용될 경우, 오류가 발생합니다(SQLSTATE 56092).

주

- USE 특권을 권한 취소한다고 해서 해당 테이블 공간에 테이블을 작성할 수 있는 능력이 취소되는 것은 아닙니다. PUBLIC 또는 그룹이 USE 특권을 보유하거나 DBADM과 같은 상위 레벨 권한을 가지는 경우, 사용자는 여전히 테이블을 해당 테이블 공간에 작성할 수 있습니다.

예

예 1: 테이블을 테이블 공간 PLANS에 작성할 수 있는 특권을 사용자 BOBBY로부터 취소하십시오.

```
REVOKE USE OF TABLESPACE PLANS FROM USER BOBBY
```

ROLLBACK

ROLLBACK문은 작업 단위(UOW) 또는 세이브포인트내에서 작성된 데이터베이스 변경사항을 복원하는 데 사용됩니다.

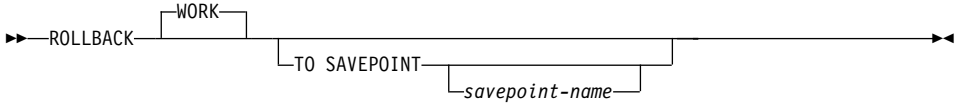
호출

이 명령문은 응용프로그램에 Embedded SQL문이나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

필요 사항 없음.

구문



설명

ROLLBACK문이 수행되는 작업 단위(UOW)는 종료되고 새로운 작업 단위가 시작됩니다. 작업 단위 중에 작성된 모든 데이터베이스 변경사항이 복원됩니다.

그러나, 다음 명령문은 트랜잭션 제어하에 있지 않으며 이들에 의한 변경사항은 ROLLBACK문의 실행과는 무관합니다.

- SET CONNECTION,
- SET CURRENT DEGREE,
- SET CURRENT DEFAULT TRANSFORM GROUP,
- SET CURRENT EXPLAIN MODE,
- SET CURRENT EXPLAIN SNAPSHOT,
- SET CURRENT PACKAGESET,
- SET CURRENT QUERY OPTIMIZATION,
- SET CURRENT REFRESH AGE,
- SET EVENT MONITOR STATE,

- SET PASSTHRU,
- SET PATH,
- SET SCHEMA,
- SET SERVER OPTION.

TO SAVEPOINT

부분적인 구간 복원(ROLLBACK TO SAVEPOINT)이 수행됨을 나타냅니다. 사용 중인 세이브포인트가 없는 경우에는 SQL 오류가 리턴됩니다(SQLSTATE 3B502). 성공적인 ROLLBACK 이후에는 세이브포인트가 계속 존재합니다. *savepoint-name*이 제공되지 않을 경우 구간 복원은 가장 최근에 설정된 세이브포인트입니다.

이 절이 생략되면 ROLLBACK WORK문이 트랜잭션 전체를 구간 복원합니다. 더우기 트랜잭션내의 세이브포인트가 해제됩니다.

savepoint-name

구간 복원될 세이브포인트를 나타냅니다. 성공적인 ROLLBACK 이후에는 *savepoint-name*으로 정의된 세이브포인트가 계속 존재합니다. 세이브포인트 이름이 없는 경우에는 오류가 리턴됩니다(SQLSTATE 3B001). 세이브포인트 설정 이후에 작성된 데이터 및 스키마 변경사항은 실행 취소됩니다.

주

- 모든 보류된 잠금은 작업 단위의 ROLLBACK시 해제됩니다. 열린 커서는 모두 닫힙니다. LOB 위치 지정자는 모두 없어집니다.
- ROLLBACK문을 실행해도 특수 레지스터 값을 변경하는 SET문이나 RELEASE문은 영향을 받지 않습니다.
- 프로그램이 비정상적으로 종료되면 작업 단위가 내재적으로 구간 복원됩니다.
- 명령문 캐쉬는 구간 복원 조작의 영향을 받습니다. 1017 페이지의 『주』에서 자세한 정보를 참조하십시오.
- 최소 단위 복합 텍스트 명령문 및 트리거와 같은 최소 단위 실행 문맥에서는 세이브포인트가 허용되지 않습니다.
- ROLLBACK TO SAVEPOINT가 커서에 미치는 영향은 세이브포인트내의 명령문에 따라 달라집니다.

  - 세이브포인트에 커서가 종속되는 DDL이 포함되는 경우 커서는 유효하지 않음으로 표시됩니다. 그러한 커서를 사용하려 하면 오류가 발생합니다 (SQLSTATE 57007).
  - 그렇지 않을 경우:
    - 커서가 세이브포인트내에서 참조되는 경우, 커서는 여전히 열려 있고 결과 테이블의 다음 논리 행 앞에 위치합니다.[104]
    - 그렇지 않으면 커서가 ROLLBACK TO SAVEPOINT의 영향을 받지 않습니다(커서는 여전히 열려 있고 위치지정됩니다).
- 동적으로 준비된 명령문 이름은 명령문을 내재적으로 다시 준비할 수 있는 경우에도 세이브포인트내에서 구간 복원되는 DDL 조작의 결과로서 여전히 유효합니다.
- ROLLBACK TO SAVEPOINT 조작은 세이브포인트내에서 명명된 모든 선언된 임시 테이블을 삭제합니다. 선언된 임시 테이블이 세이브포인트내에서 수정되는 경우 해당 테이블의 모든 행이 삭제됩니다.
- ROLLBACK TO SAVEPOINT문 다음의 모든 잠금은 보존됩니다.
- ROLLBACK TO SAVEPOINT 조작 다음의 모든 LOB 위치 지정자는 보존됩니다.

예

최종 확약점 또는 구간 복원 이후 변경된 사항을 삭제합니다.

```
ROLLBACK WORK
```

104. 위치지정된 UPDATE 또는 DELETE문을 발행하기 전에 FETCH를 수행해야 합니다.

SAVEPOINT

SAVEPOINT 문을 사용하여 트랜잭션내에서 세이브포인트를 설정하십시오.

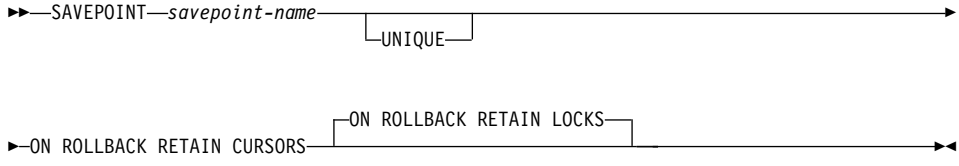
호출

이 명령문은 응용프로그램(저장 프로시듀어 포함)에 넣거나 대화식으로 발행할 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

필요사항 없음.

구문



설명

savepoint-name

*savepoint*의 이름입니다.

UNIQUE

UNIQUE 세이브포인트 지정은 응용프로그램이 세이브포인트 사용시 이 세이브포인트 이름을 재사용하지 않으려 함을 나타냅니다.

ON ROLLBACK RETAIN CURSORS

SAVEPOINT 문 다음에 처리되는 커서 명령문을 열기 위해 이 세이브포인트로 구간 복원할 때의 시스템 동작을 지정합니다. RETAIN CURSORS 절은 가능할 때마다 세이브포인트로의 구간 복원에 의해 커서가 변경되지 않음을 나타냅니다. 세이브포인트로의 구간 복원이 커서에 영향을 주는 상황인 경우에는 1130 페이지의 『ROLLBACK』을 참조하십시오.

ON ROLLBACK RETAIN LOCKS

세이브포인트 설정 이후에 획득되는 잠금에 대해 이 세이브포인트로 구간 복

원할 때의 시스템 동작을 지정합니다. 세이트포인트 이후 획득된 잠금이 세이브포인트로의 구간 복원시 추적되지 않고 구간 복원(해제)되지 않습니다.

규칙

- 세이브포인트를 중첩시킬 수 없습니다. 세이브포인트 명령문이 발행되고 이미 설정된 세이브포인트가 있는 경우에는 오류가 발생합니다(SQLSTATE 3B002).

주

- OS/390용 DB2 Universal Database와의 호환성을 위해 UNIQUE 키워드가 지원되었습니다. 다음은 OS/390용 DB2 Universal Database에서의 동작을 설명합니다.

  세이브포인트 이름 *savepoint-name*이 이미 트랜잭션내에 있으면 오류가 리턴됩니다(SQLSTATE 3B501). 응용프로그램은 UNIQUE 고유을 생략함으로써 이 세이브포인트 이름이 해당 트랜잭션내에서 다시 사용될 수도 있다고 주장합니다. *savepoint-name*이 이미 트랜잭션에 있는 경우, 이 이름은 없어지고 새 세이브포인트 *savepoint-name*이 작성됩니다.

  다른 세이브포인트에 대한 이름을 다시 사용함으로써 세이브포인트를 없애는 작업은 RELEASE SAVEPOINT 문으로 이전 세이브포인트를 해제하는 것과 같지 않습니다. 이름 재사용에 의한 세이브포인트 없애기는 해당 세이브포인트만을 없앱니다. RELEASE SAVEPOINT 문으로 세이브포인트를 해제하면 명명된 세이브포인트와 명명된 세이브포인트 이후에 설정된 모든 세이브포인트가 해제됩니다.
- 세이브포인트내에서 유틸리티, SQL문 또는 DB2 명령이 처리 중에 중간 COMMIT 문을 수행하면, 세이브포인트가 내재적으로 해제됩니다.
- SQL문 SET INTEGRITY는 세이브포인트내에서의 DDL문과 같은 효과를 가집니다.
- 응용프로그램에서 삽입은 버퍼화될 수 있습니다(즉, 응용프로그램이 INSERT BUF 옵션으로 사전 처리 컴파일되었습니다). SAVEPOINT, ROLLBACK 또는 RELEASE TO SAVEPOINT문이 발행되면 버퍼가 비워집니다.

SELECT

SELECT문은 조회의 한 형태입니다. 이는 응용프로그램에 포함되거나, 대화식으로 발행될 수 있습니다. 482 페이지의『select문』 및 434 페이지의『부속 선택』에서 자세한 정보를 참조하십시오.

SELECT INTO

SELECT INTO문은 많아야 한 행으로 구성되는 결과 테이블을 작성한 후, 그 행의 값을 호스트 변수에 지정합니다. 테이블이 공백인 경우, 명령문은 +100을 SQLCODE에, '02000'을 SQLSTATE에 지정하고, 호스트 변수에는 값을 지정하지 않습니다. 하나 이상의 행이 검색 조건을 충족시킬 경우, 명령문 처리는 종료되고 오류가 발생합니다(SQLSTATE 21000).

호출

이 명령문은 응용프로그램에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

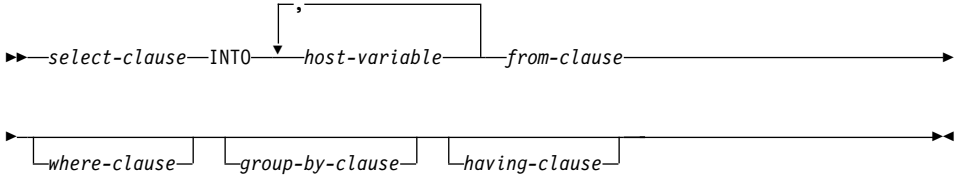
권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권에는 SELECT INTO문에서 참조되는 각 테이블 또는 뷰에 대해 적어도 다음 중 하나가 포함되어야 합니다.

- SELECT 특권
- CONTROL 특권
- SYSADM 또는 DBADM 권한

SELECT INTO문에 대해서는 GROUP 특권이 점검되지 않습니다.

구문



설명

433 페이지의 『제5장 조회』에 *select-clause*, *from-clause*, *where-clause*, *group-by-clause* 및 *having-clause*에 대한 설명을 참조하십시오.

INTO

호스트 변수 목록을 나열합니다.

host-variable

호스트 변수 선언 규칙에 따른 프로그램에 기술된 변수를 식별합니다.

결과 행의 첫번째 값은 목록의 첫번째 변수에 지정되고, 두 번째 값은 두 번째 호스트 변수에, 이와 같은 식으로 지정됩니다. 호스트 변수의 수가 컬럼 값보다 작으면 'W' 값이 SQLCA의 SQLWARN3 필드에 지정됩니다 (1261 페이지의 『부록B. SQL 통신(SQLCA)』 참조).

109 페이지의 『지정 및 비교』에 설명되어 있는 규칙에 따라 변수가 각각 지정됩니다. 목록을 통해 순서대로 지정됩니다.

오류가 발생하면, 값이 호스트 변수에 지정되지 않습니다.

예

예 1: 이 C 예는 EMP의 최대 급여를 호스트 변수 MAXSALARY에 둡니다.

```
EXEC SQL SELECT MAX(SALARY)
  INTO :MAXSALARY
  FROM EMP;
```

예 2: 이 C 예는 EMP의 사원 528671에 대한 행을 호스트 변수에 둡니다.

```
EXEC SQL SELECT * INTO :h1, :h2, :h3, :h4
  FROM EMP
  WHERE EMPNO = '528671';
```

SET CONNECTION

SET CONNECTION문은 휴면 상태인 연결을 현재 상태로 변경하여, 지정된 위치를 현재 서버로 만듭니다. 이는 트랜잭션 제어하에 있지 않습니다.

호출

대화식 SQL 기능에서 대화식 실행의 형태를 나타내는 인터페이스를 제공하더라도, 이 명령문은 응용프로그램 내에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

필요사항 없음.

구문

설명

server-name 또는 *host-variable*

*server-name*이 들어 있는 *host-variable* 또는 지정된 *server-name*으로 응용프로그램 서버(AS)를 식별합니다.

*host-variable*가 지정되면, 길이가 8자 이하인 문자열 변수이어야 하며 표시기 변수를 포함해서는 안 됩니다. *host-variable* 내에 들어 있는 *server-name*은 왼쪽으로 정렬되어야 하고, 따옴표로 구분해서는 안 됩니다.

*server-name*은 응용프로그램 서버(AS)를 식별하는 데이터베이스 별명임에 유의하십시오. 응용프로그램 리퀘스터의 지역 디렉토리에 나열되어야 합니다.

server-name 또는 *host-variable*는 응용프로그램 프로세스의 기존 연결을 식별해야 합니다. 기존의 연결을 식별하지 않는 경우, 오류(SQLSTATE 08003)가 발생합니다.

SET CONNECTION이 현재 연결로 되면, 응용프로그램 프로세스의 모든 연결 상태는 변경되지 않습니다.

성공적인 연결:

SET CONNECTION문이 성공적으로 실행되면,

- 연결이 작성되지 않습니다. CURRENT SERVER 특수 레지스터는 지정된 *server-name*으로 갱신됩니다.
- 이전의 현재 연결이 존재하면, 비활동 상태(다른 *server-name*이 지정된 것으로 가정)로 지정됩니다.
- CURRENT SERVER 특수 레지스터와 SQLCA는 유형 1 CONNECT에 기술된 것과 동일한 방법으로 갱신됩니다. 자세한 사항은 611에 있습니다.

실패한 연결:

SET CONNECTION문이 실패하는 경우,

- 실패한 이유가 무엇이든지, 응용프로그램 프로세스의 연결 상태 및 연결 상태는 변경되지 않습니다.
- 실패한 유형 1 CONNECT과 함께, SQLCA의 SQLERRP 필드는 오류를 검출한 모듈의 이름으로 설정됩니다.

주

- CONNECT 유형 1문을 사용해도 SET CONNECTION을 사용할 수는 있지만, SET CONNECTION문이 현재 연결을 지정하지 않는 한, 휴면 상태의 연결이 존재할 수 없으므로 명령문은 항상 실패합니다(SQLSTATE 08003).
- SQLRULES(DB2) 연결 옵션(45 페이지의 『분산 작업 단위(DUOW) 의미 정의 옵션』 참조)은 SET CONNECTION을 사용하지 못하게 하는 것은 아니지만, CONNECT 유형 2문이 대신 사용될 수 있으므로 필요하지 않습니다.
- 연결이 사용되고, 휴면 상태로 된 후 같은 작업 단위(UOW) 내에서 현재 상태로 복원되면, 이 연결은 잠금, 커서 및 준비된 명령문 상태와 관련하여 응용프로그램 프로세스에 의해 최종 사용됨을 반영합니다.

예

IBMSTHDB에서 SQL문을 실행하고, IBMTOKDB에서 SQL문을 실행한 후 IBMSTHDB에서 더 많은 SQL문을 실행합니다.

```
EXEC SQL CONNECT TO IBMSTHDB;
/* Execute statements referencing objects at IBMSTHDB */
```

```
EXEC SQL CONNECT TO IBMTOKDB;
/* Execute statements referencing objects at IBMTOKDB */

EXEC SQL SET CONNECTION IBMSTHDB;
/* Execute statements referencing objects at IBMSTHDB */
```

첫번째 CONNECT문이 IBMSTHDB 연결을 하고, 두 번째 CONNECT문은 이를 휴면 상태로 두고, SET CONNECTION문은 이를 다시 현재 상태로 리턴함에 유의하십시오.

SET CURRENT DEFAULT TRANSFORM GROUP

SET CURRENT DEFAULT TRANSFORM GROUP문은 CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터 값을 변경합니다. 이 명령문은 트랜잭션 제어하에 있습니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문을 실행하는 데 있어 권한은 필요없습니다.

구문

설명

group-name

모든 구조화 유형에 대해 정의된 변환 그룹을 식별하는 한 부분 이름을 지정합니다. 이 이름은 후속 명령문에서 참조할 수 있습니다(또는 다른 SET CURRENT DEFAULT TRANSFORM GROUP 문을 사용하여 다시 특수 레지스터 값을 변경할 때까지).

이 이름은 최대 18자의 SQL 식별자이어야 합니다(SQLSTATE 42815). 특수 레지스터가 설정될 때까지는 모든 구조화 유형의 *group-name*이 정의되었는지에 대한 유효성 검사가 수행되지 않습니다. 구조화 유형이 특별히 참조되는 경우에만 유효성 검사를 위해 명명된 변환 그룹 정의가 확인됩니다.

규칙

- 지정 값이 *group-name*에 대한 규칙을 지키지 않으면, 오류가 발생합니다 (SQLSTATE 42815).

- *group-name* 변환 그룹에 정의된 TO SQL 및 FROM SQL 함수는 사용자 정의 구조화 유형을 호스트 프로그램과 교환하는 데 사용됩니다.

주

- CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터의 초기 값은 빈 스트링입니다.
- 특수 레지스터의 사용에 관한 추가 규칙에 대해서는 137 페이지의 『CURRENT DEFAULT TRANSFORM GROUP』을 참조하십시오.

예

예 1: 기본 변환 그룹을 MYSTRUCT1으로 설정하십시오. MYSTRUCT1 변환 그룹에 정의된 TO SQL 및 FROM SQL 함수는 사용자 정의 구조화 유형 변수를 현재 호스트 프로그램과 교환하는 데 사용됩니다.

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

SET CURRENT DEGREE

SET CURRENT DEGREE문은 값을 CURRENT DEGREE 특수 레지스터에 지정합니다. 이 명령문은 트랜잭션 제어하에 있습니다.

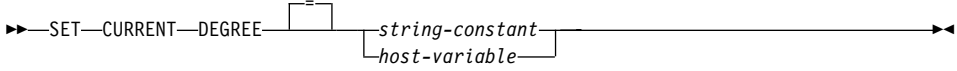
호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문을 실행하는 데 있어 권한은 필요없습니다.

구문



설명

CURRENT DEGREE의 값은 문자열 산수나 호스트 변수의 값으로 대체됩니다. 값은 5바이트를 넘지 않는 문자열이어야 합니다. 값은 1에서 32 767까지의 정수 문자열 표현이거나 'ANY'이어야 합니다.

SQL문이 동적으로 준비되고 정수로 표시된 CURRENT DEGREE의 값이 1인 경우, 이 명령문의 실행은 파티션 내 병렬 처리를 사용하지 않습니다.

SQL문이 동적으로 준비될 때 CURRENT DEGREE의 값이 번호이면, 그 명령문의 실행이 지정된 수준으로 파티션 내 병렬 처리가 수반됩니다.

SQL문이 동적으로 준비되고 CURRENT DEGREE의 값이 'ANY'인 경우, 이 명령문의 실행에는 데이터베이스 관리 프로그램이 정한 수준으로 파티션 내 병렬 처리가 관련됩니다.

host-variable

*host-variable*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하고, 길이는 5자를 초과할 수 없습니다. 더 긴 필드가 제공되면 오류가 리턴됩니다 (SQLSTATE 42815). 제공된 실제 값이 지정된 대체 값보다 크면, 공백을 입

력하여 오른쪽을 채워야 합니다. 선행 공백은 허용되지 않습니다(SQLSTATE 42815). 모든 입력값은 대소문자를 구별합니다. *host-variable*에 관련된 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(NULL) 값을 표시해서는 안 됩니다(SQLSTATE 42815).

string-constant

string-constant 길이는 5를 초과할 수 없습니다.

주

정적 SQL문의 파티션 내 병렬 처리 수준은 PREP 또는 BIND 명령의 DEGREE 옵션을 사용하여 제어할 수 있습니다. 이들 명령에 대한 *Command Reference*에서 자세한 내용을 참조하십시오.

파티션 내 병렬 처리의 실제 수행시 수준은 다음 수준보다 낮습니다.

- 최대 조회 수준(max_querydegree) 구성 매개변수
- 응용프로그램 런타임 수준
- SQL문 컴파일 수준

파티션 내 병렬 처리를 사용하려면 intra_parallel 데이터베이스 관리 프로그램 구성이 on이어야 합니다. off로 설정된 경우, 이 레지스터의 값은 무시되며 명령문은 최적화를 위해 파티션 내 병렬 처리를 사용하지 않습니다(SQLSTATE 01623).

일부 SQL문은 파티션 내 병렬 처리를 사용할 수 없습니다. 파티션 내 병렬 처리 수준과 제한사항 목록을 보려면 *관리 안내서*에서 자세한 설명을 참조하십시오.

예

예 1: 다음 명령문은 CURRENT DEGREE를 설정하여 파티션 내 병렬 처리를 금지합니다.

```
SET CURRENT DEGREE = '1'
```

예 2: 다음 명령문은 CURRENT DEGREE를 설정하여 파티션 내 병렬 처리를 허용합니다.

```
SET CURRENT DEGREE = 'ANY'
```

SET CURRENT EXPLAIN MODE

SET CURRENT EXPLAIN MODE문은 CURRENT EXPLAIN MODE 특수 레지스터 값을 변경합니다. 이는 트랜잭션 제어하에 있지 않습니다.

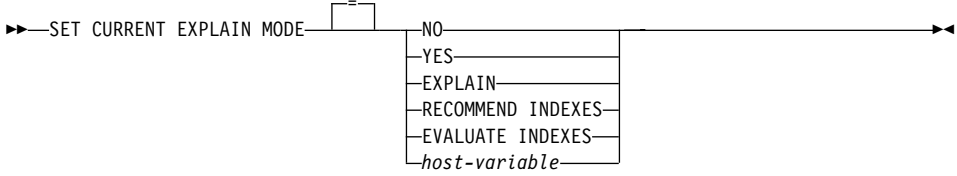
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문을 실행하는 데 특수 권한은 필요없습니다.

구문



설명

NO

'Explain' 기능을 작동 불가능으로 만듭니다. 캡쳐된 Explain 정보가 없습니다. NO는 특수 레지스터의 초기값입니다.

YES

Explain 기능을 작동 가능으로 하고 Explain 정보가 적합한 동적 SQL문에 대한 Explain 테이블에 삽입되도록 합니다. 모든 동적 SQL문이 컴파일되고 정상적으로 실행됩니다.

EXPLAIN

Explain 기능을 작동가능으로 하고 Explain 정보가 준비된 동적 SQL문에 대해 캡쳐되도록 합니다. 그러나 동적 명령문은 실행되지 않습니다.

RECOMMEND INDEXES

SQL 컴파일러가 색인을 권고할 수 있게 합니다. 이 설명 모드에서 실행되는

모든 조회는 ADVISE_INDEX 테이블을 권장되는 색인으로 채울 것입니다. 또한, 권장되는 색인이 어떻게 사용되는지를 보이기 위해 설명 테이블에서 설명 정보가 캡쳐될 것이지만, 명령문은 컴파일되지도 실행되지도 않습니다.

EVALUATE INDEXES

SQL 컴파일러가 색인을 평가할 수 있게 합니다. 평가될 색인을 ADVISE_INDEX 테이블로부터 읽어 EVALUATE = Y로 표시해야 합니다. 최적화 알고리즘은 카탈로그로부터의 값에 기초하여 가상 색인을 생성합니다. 이 설명 모드에서 실행되는 모든 조회는 가상 색인에 기초하여 예측된 통계를 사용하여 컴파일되고 최적화될 것입니다. 명령문은 실행되지 않습니다.

host-variable

*host-variable*은 데이터 유형이 CHAR 또는 VARCHAR여야 하고, 길이는 254 자를 초과할 수 없습니다. 이보다 긴 필드가 지정되면 오류(SQLSTATE 42815)가 리턴됩니다. 지정된 값은 NO, YES, EXPLAIN, RECOMMEND INDEXES 또는 EVALUATE INDEXES이어야 합니다. 제공된 실제 값이 지정된 대체 값보다 크면, 공백을 입력하여 오른쪽을 채워야 합니다. 선행 공백은 허용되지 않습니다(SQLSTATE 42815). 모든 입력값은 대소문자를 구별합니다. *host-variable*에 관련된 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(NULL) 값을 표시해서는 안 됩니다(SQLSTATE 42815).

주

정적 SQL문에 대한 Explain 정보는 PREP 또는 BIND 명령의 EXPLAIN 옵션을 사용하여 보관될 수 있습니다. EXPLAIN 옵션의 ALL값이 지정되고 CURRENT EXPLAIN MODE 레지스터 값이 NO이면, Explain 정보가 런타임 동적 SQL문에 의해 보관됩니다. CURRENT EXPLAIN MODE 레지스터의 값이 NO가 아니면, EXPLAIN 바인드 옵션값이 무시됩니다. EXPLAIN 옵션과 CURRENT EXPLAIN MODE 특수 레지스터 사이의 상호 작용에 대한 1490 페이지의 표143에서 자세한 정보를 참조하십시오.

RECOMMEND INDEXES 및 EVALUATE INDEXES는 SET CURRENT EXPLAIN MODE 명령으로만 설정될 수 있는 특수 모드입니다. 이들 모드는 PREP나 BIND 옵션을 사용하여 설정될 수 없으며, SET CURRENT SNAPSHOT 명령과 작동하지 않습니다.

Explain 기능이 활성화되면, 현재 권한 부여 ID는 Explain 테이블의 INSERT 특권이 있어야 하거나 또는 오류(SQLSTATE 42501)가 발생합니다.

*관리 안내서*에서 자세한 정보를 참조하십시오.

예

예 1: 다음 명령문은 적절한 후속 동적 SQL문에 대한 Explain 정보가 캡쳐되고 명령문이 실행되지 않도록 CURRENT EXPLAIN MODE 특수 레지스터를 설정합니다.

```
SET CURRENT EXPLAIN MODE = EXPLAIN
```

SET CURRENT EXPLAIN SNAPSHOT

SET CURRENT EXPLAIN SNAPSHOT문은 CURRENT EXPLAIN SNAPSHOT 특수 레지스터 값을 변경합니다. 이는 트랜잭션 제어하에 있지 않습니다.

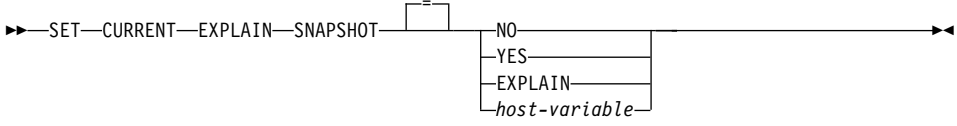
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문을 실행하는 데 있어 권한은 필요없습니다.

구문



설명

NO

Explain 스냅샷 기능을 사용할 수 없습니다. 스냅샷이 구해지지 않았습니다. NO는 특수 레지스터의 초기값입니다.

YES

Explain 스냅샷 기능을 작동 가능으로 하고, 적합한 각 동적 SQL문에 대한 내부 표현의 스냅샷을 작성합니다. 이 정보는 EXPLAIN_STATEMENT 테이블의 SNAPSHOT 컬럼에 삽입됩니다(1455 페이지의 『부록K. Explain 테이블 및 정의』 참조).

EXPLAIN SNAPSHOT 기능은 Visual Explain에 사용하기 위한 것입니다.

EXPLAIN

Explain 스냅샷 기능을 작동시키고, 적합한 각 동적 SQL문에 대한 내부 표현의 스냅샷을 작성합니다. 그러나 동적 명령문은 실행되지 않습니다.

host-variable

*host-variable*의 데이터 유형은 CHAR 또는 VARCHAR이어야 하고, 그 내용의 길이는 8자를 초과할 수 없습니다. 더 긴 필드가 제공되면 오류가 리턴됩니다(SQLSTATE 42815). 이 레지스터에 들어 있는 값은 NO, YES 또는 EXPLAIN이어야 합니다. 제공된 실제 값이 지정된 대체 값보다 크면, 공백을 입력하여 오른쪽을 채워야 합니다. 선행 공백은 허용되지 않습니다(SQLSTATE 42815). 모든 입력값은 대소문자를 구별합니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(NULL) 값을 표시해서는 안 됩니다(SQLSTATE 42815).

주

정적 SQL문에 대한 Explain 스냅샷이 PREP 또는 BIND 명령의 EXPLSNAP 옵션을 사용하여 보관될 수 있습니다. EXPLSNAP 옵션의 ALL값이 지정되고 CURRENT EXPLAIN SNAPSHOT 레지스터 값이 NO이면, Explain 스냅샷은 런타임 동적 SQL문에 대해 보관됩니다. CURRENT EXPLAIN SNAPSHOT 레지스터의 값이 NO가 아니면, EXPLSNAP 옵션이 무시됩니다. EXPLSNAP 옵션과 CURRENT EXPLAIN SNAPSHOT 특수 레지스터 사이의 상호 작용에 대한 1491 페이지의 표144에서 자세한 정보를 참조하십시오.

Explain 스냅샷 기능이 활성화되면, 현재 권한 부여 ID는 Explain 테이블에 대한 INSERT 특권이 있어야 하거나 또는 오류(SQLSTATE 42501)가 발생합니다.

*관리 안내서*에서 자세한 정보를 참조하십시오.

예

예 1: 다음 명령문은 CURRENT EXPLAIN SNAPSHOT 특수 레지스터를 설정하므로, 후속하는 적절한 동적 SQL문에 대해 Explain 스냅샷이 취해지고 명령문이 실행됩니다.

```
SET CURRENT EXPLAIN SNAPSHOT = YES
```

예 2: 다음 예는 CURRENT EXPLAIN SNAPSHOT 특수 레지스터의 현재 값을 SNAP이라는 호스트 변수로 검색합니다.

```
EXEC SQL VALUES (CURRENT EXPLAIN SNAPSHOT) INTO :SNAP;
```

SET CURRENT PACKAGESET

SET CURRENT PACKAGESET문은 후속 SQL문에 사용할 패키지를 선택하는 데 사용할 스키마 이름(콜렉션 식별자)을 설정합니다. 이 명령문은 트랜잭션 제어 하에 있습니다.

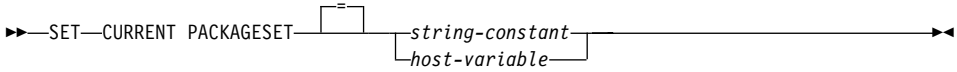
호출

이 명령문은 응용프로그램에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다. 이 명령문은 REXX에서 지원되지 않습니다.

권한 부여

필요 사항 없음.

구문



설명

string-constant

최대 30자 길이의 문자열 상수. 최대 길이를 초과하면, 런타임시 절단됩니다.

host-variable

최대 30자 길이의 유형 CHAR 또는 VARCHAR의 변수. 이는 널(NULL)로 설정될 수 없습니다. 최대 길이를 초과하면, 런타임시 절단됩니다.

주

- 이 명령문으로 응용프로그램은 실행 가능 SQL문에 패키지를 선택할 때 사용되는 스키마 이름을 지정할 수 있습니다. 명령문은 클라이언트에서 처리되고, 응용프로그램 서버(AS)로 가지 않습니다.
- COLLECTION 바인드 옵션은 지정된 스키마 이름으로 패키지를 작성하는 데 사용할 수 있습니다. *Command Reference*에서 자세한 내용을 참조하십시오.
- MVS/ESA용 DB2와는 달리, SET CURRENT PACKAGESET문은 CURRENT PACKAGESET라고 하는 특수 레지스터의 지원없이 구현됩니다.

예

TRYIT이라는 응용프로그램이 사용자 id PRODUSA(이는 바인드 파일에 기본 스키마 이름 'PRODUSA'를 만듭니다)로 사전 처리 컴파일되었다고 가정합니다. 응용프로그램은 다른 바인드 옵션으로 두번 바인드됩니다. 다음과 같은 명령행 처리기 명령이 사용되었습니다.

```
DB2 CONNECT TO SAMPLE USER PRODUSA
DB2 BIND TRYIT.BND DATETIME USA
DB2 CONNECT TO SAMPLE USER PRODEUR
DB2 BIND TRYIT.BND DATETIME EUR COLLECTION 'PRODEUR'
```

이는 TRYIT라는 두 개의 패키지 파일을 작성합니다. 첫번째 바인드 명령은 'PRODUSA'라는 스키마에 패키지를 작성하였습니다. 두 번째 바인드 명령은 COLLECTION 옵션에 기초하여 'PRODEUR'라는 스키마에 패키지를 작성하였습니다.

응용프로그램 TRYIT에 다음의 명령문이 들어 있다고 가정합니다.

```
EXEC SQL CONNECT TO SAMPLE;
 .
 .
EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 1
 .
 .
EXEC SQL SET CURRENT PACKAGESET 'PRODEUR';                            2
 .
 .
EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 3
```

1 이 명령문은 PRODUSA.TRYIT 패키지를 사용하여 수행하게 되는데, 이것이 응용프로그램에 대한 기본 패키지이기 때문입니다. 그러므로, 날짜는 USA 형식으로 리턴됩니다.

2 이 명령문은 패키지 선택에 대해 스키마 이름을 'PRODEUR'에 설정합니다.

3 이 명령문은 SET CURRENT PACKAGESET문의 결과로 PRODEUR.TRYIT 패키지를 사용하여 수행됩니다. 그러므로, 날짜는 EUR 형식으로 리턴됩니다.

SET CURRENT QUERY OPTIMIZATION

SET CURRENT QUERY OPTIMIZATION문은 CURRENT QUERY OPTIMIZATION 특수 레지스터에 값을 지정합니다. 이 값은 동적 SQL문을 준비할 때 작동 가능한 현재 최적화 기법 클래스를 지정합니다. 이는 트랜잭션 제어 하에 있지 않습니다.

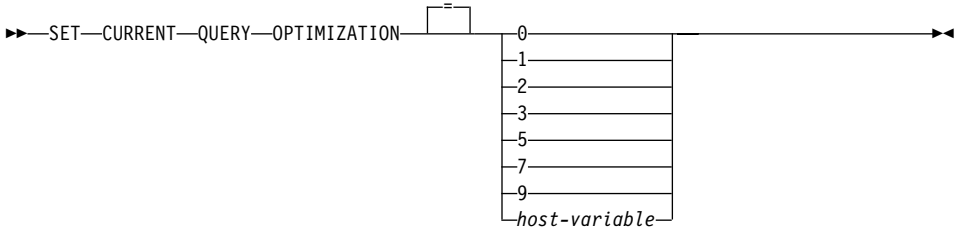
호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문을 실행하는 데 있어 권한은 필요없습니다.

구문



설명

optimization-class

*optimization-class*는 정수 상수 또는 실행시 해당값이 포함될 호스트 변수 이름으로서 지정될 수 있습니다. 다음은 클래스에 대한 개요입니다.(*관리 안내서*에서 자세한 내용을 참조하십시오.)

0 액세스 플랜을 생성하는 데 최소 최적화가 수행되도록 지정합니다. 이 클래스는 색인이 잘 구성된 테이블을 단순한 동적 SQL로 액세스하는 데 가장 적합합니다.

1 액세스 플랜을 생성하는 데 대략적으로 DB2 버전 1에 비교되는 최적화가 수행되도록 지정합니다.

2 DB2 버전 1의 최적화 레벨보다 높게 레벨을 지정합니다. 그러나 이것은 레벨 3보다 낮은 최적화 비용에서 지정되며, 특히 매우 복잡한 조회들에 대한 것입니다.

3 액세스 플랜 생성을 위해 적당한 양의 최적화가 수행되도록 지정합니다.

5 액세스 플랜 생성을 위해 많은 양의 최적화가 수행되도록 지정합니다. 복잡한 동적 SQL 조회의 경우, 액세스 플랜 선택에 소비하는 시간량을 제한하는 데 경험적 규칙이 사용됩니다. 가능한 경우, 조회는 기본이 되는 기본 테이블 대신 요약 테이블을 사용합니다.

7 액세스 플랜 생성을 위해 많은 양의 최적화가 수행되도록 지정합니다. 5와 유사하지만, 경험적 규칙이 없습니다.

9 액세스 플랜 수행을 위해 최대 최적화가 수행되도록 지정합니다. 이는 평가될 수 있는 액세스 플랜의 수를 확대합니다. 이 클래스는 큰 테이블을 사용하여 매우 복잡하고 장시간 실행되는 조회에 대해 더 나은 액세스 플랜이 생성될 수 있는지를 결정하는 데 사용되어야 합니다. Explain 및 성능 측정을 사용하여 더 나은 플랜이 생성되었는지를 검증할 수 있습니다.

host-variable 데이터 유형은 INTEGER입니다. 값은 0 - 9 범위에 있어야 하는데(SQLSTATE 42815), 이는 0, 1, 2, 3, 5, 7 또는 9 여야 합니다(SQLSTATE 01608). *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(NULL) 값을 표시해서는 안 됩니다(SQLSTATE 42815).

주

- CURRENT QUERY OPTIMIZATION 레지스터가 특정 값으로 설정되면, 조회 재작성 규칙 세트가 작동 가능하게 되고, 일부 최적화 변수는 특정 값을 취합니다. 이러한 최적화 기법 클래스는 동적 SQL문 준비중에 사용됩니다.
- 일반적으로, 최적화 클래스를 변경하면, 응용프로그램의 실행 시간, 컴파일 시간, 필요 자원 등이 영향을 받습니다. 대부분의 명령문은 기본 조회 최적화 클래스를 사용하여 적절하게 최적화됩니다. 낮은 조회 최적화 클래스, 특히 클래스1과

클래스2는 동적 *PREPARE*에 의해 소비되는 자원이 조회 실행에 필요한 자원의 중요 부분인 동적 SQL문에 적절합니다. 높은 최적화 클래스는 소비될 추가 자원을 고려하고 더 나은 액세스 플랜이 생성된 이후에만 선택할 수 있습니다. 각 조회 최적화 클래스와 연관된 작동 방식에 대한 *관리 안내서*에서 자세한 내용을 참조하십시오.

- 조회 최적화 클래스는 범위 0-9에 속해야 합니다. 이 범위 밖의 클래스는 오류를 리턴합니다(SQLSTATE 42815). 이 범위 내에서 지원되지 않는 클래스는 경고를 리턴하고(SQLSTATE 01608) 다음으로 낮은 조회 최적화 클래스로 교체됩니다. 예를 들면, 조회 최적화 클래스 6은 5로 대체됩니다.
- 동적으로 준비된 명령문은 가장 최근에 실행된 SET CURRENT QUERY OPTIMIZATION문에 의해 설정된 최적화 클래스를 사용합니다. SET CURRENT QUERY OPTIMIZATION문이 아직 실행되지 않은 경우, 조회 최적화 클래스는 데이터베이스 구성 매개변수 dft_queryopt의 값으로 결정됩니다.
- 통계적으로 바인드된 명령문은 CURRENT QUERY OPTIMIZATION 특수 레지스터를 사용하지 않으므로, 이 명령문은 아무 영향을 미치지 않습니다. QUERYOPT 옵션은 사전 처리 또는 바인드시 통계적으로 바인드된 명령문에 대해 원하는 최적화 클래스를 지정하는 데 사용됩니다. QUERYOPT가 지정되지 않은 경우, 데이터베이스 구성 매개변수 dft_queryopt가 지정하는 기본값이 사용됩니다. BIND 명령에 대해서는 *Command Reference*에서 세부사항을 참조하십시오.
- SET CURRENT QUERY OPTIMIZATION문이 실행되는 작업 단위(UOW)가 구간 복원된 경우, 이 명령문의 실행 결과는 구간 복원되지 않습니다.

예

예 1: 이 예는 가장 높은 수준의 최적화가 어떻게 선택되는지를 보여줍니다.

```
SET CURRENT QUERY OPTIMIZATION 9
```

예 2: 다음 예는 조회 내에서 CURRENT QUERY OPTIMIZATION 특수 레지스터의 사용 방식을 보여줍니다.

SYSCAT.PACKEGES 카탈로그 값을 사용하여, CURRENT QUERY OPTIMIZATION 특수 레지스터의 현재 값과 같은 설정치로 제한된 모든 계획을 찾습니다.

```
EXEC SQL  DECLARE C1 CURSOR FOR
  SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
  WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

SET CURRENT REFRESH AGE

SET CURRENT REFRESH AGE문은 CURRENT REFRESH AGE 특수 레지스터의 값을 변경합니다. 이는 트랜잭션 제어하에 있지 않습니다.

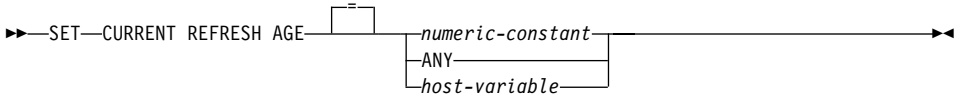
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문을 실행하는 데 있어 권한은 필요없습니다.

구문



설명

numeric-constant

DECIMAL(20,6)는 시간소인 기간을 나타냅니다. 값은 0 또는 99 999 999 999 999이어야 합니다(값 중에서 마이크로 초 부분은 무시되므로 아무 값이나 될 수 있음).

0
REFRESH IMMEDIATE가 지정된 요약 테이블만이 조회 처리를 최적화하는 데 사용될 수 있음을 나타냅니다.

99999999999999
REFRESH DEFERRED 또는 REFRESH IMMEDIATE가 지정된 요약 테이블만이 조회 처리를 최적화하는 데 사용될 수 있음을 나타냅니다. 이 값은 9 999년, 99월, 99일, 99시, 99분 및 99초를 나타냅니다.

ANY

이는 99999999999999를 줄여 표시한 것입니다.

host-variable

DECIMAL(20,6) 유형의 변수 또는 DECIMAL(20,6)에 할당된 다른 유형. 이는 널(NULL)로 설정될 수 없습니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(NULL) 값을 표시해서는 안 됩니다 (SQLSTATE 42815). 호스트 변수의 값은 0 또는 99 999 999 999 999.000000 이어야 합니다.

주

- CURRENT REFRESH AGE 특수 레지스터의 초기 값은 0입니다.
- CURRENT REFRESH AGE 특수 레지스터의 값을 0이 아닌 다른 값으로 설정하면 경고와 함께 수행됩니다. 조회 처리를 최적화하는 데 사용될기반 기본 테이블의 값을 나타내지 못할 수도 있는 요약 테이블을 허용함으로써, 조회 결과가 기반 테이블의 데이터를 정확히 나타내지 못할 수도 있습니다. 이는 기반 데이터가 변경되지 않았음을 알고 있거나 데이터에 대한 사용자의 지식을 토대로 결과에 있을 수 있는 어느 정도의 오류를 수용할 용의가 있을 경우에 적당합니다.
- CURRENT REFRESH AGE의 값 99 999 999 999 999는 결과가 유효한 날짜 범위를 벗어나기 때문에 시간소인 산술 연산에 사용될 수 없습니다 (SQLSTATE 22008).

예

예 1: 다음 명령문은 CURRENT REFRESH AGE 특수 레지스터를 설정합니다.

```
SET CURRENT REFRESH AGE ANY
```

예 2:

다음 예는 CURRENT REFRESH AGE 특수 레지스터의 현재 값을 CURMAXAGE라는 호스트 변수로 읽어들입니다.

```
EXEC SQL VALUES (CURRENT REFRESH AGE) INTO  :CURMAXAGE;
```

값은 이전 예에서 설정된 99999999999999.000000입니다.

SET EVENT MONITOR STATE

SET EVENT MONITOR STATE문은 이벤트 모니터를 활성화 또는 비활성화합니다. 이벤트 모니터의 현재 상태(활성 또는 비활성)는 EVENT_MON_STATE 내장 함수를 이용하여 알 수 있습니다. SET EVENT MONITOR STATE문은 트랜잭션 제어하에 있지 않습니다.

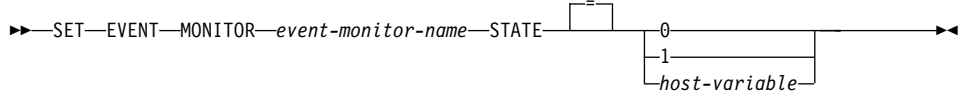
호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

명령문의 권한 부여 ID에는 SYSADM이나 DBADM 권한이 있어야 합니다(SQLSTATE 42815).

구문



설명

이벤트 모니터 이름

활성화 또는 비활성화할 이벤트 모니터를 식별합니다. 이 이름은 카탈로그에 존재하는 이벤트 모니터를 식별해야 합니다(SQLSTATE 42704).

새로운 상태

*새로운 상태*는 런타임 해당값이 들어갈 정수 상수 또는 호스트 변수 이름으로서 지정될 수 있습니다. 다음을 지정할 수 있습니다.

0 지정된 이벤트 모니터가 비활성화되어야 함을 표시합니다.

1 지정된 이벤트 모니터가 활성화되어야 함을 표시합니다. 이벤

트 모니터가 이미 활성화되어 있어서는 안 됩니다. 그렇지 않은 경우, 경고(SQLSTATE 01598)가 발행됩니다.

호스트 변수 데이터 유형은 INTEGER입니다. 지정된 값은 0 또는 1이어야 합니다(SQLSTATE 42815). *호스트 변수*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(NULL) 값을 표시해서는 안 됩니다(SQLSTATE 42815).

규칙

- 이벤트 모니터를 무제한으로 정의할 수는 있어도, 최고 32개의 이벤트 모니터만이 동시에 활동 상태에 있을 수 있습니다(SQLSTATE 54030).
- 이벤트 모니터를 활성화하려면, 이벤트 모니터가 작성된 트랜잭션이 확약되었어야 합니다(SQLSTATE 55033). 이 규칙은 (하나의 작업 단위(UOW)에서) 이벤트 모니터의 작성, 모니터 활성화 및 트랜잭션의 구간 복원을 금지합니다.
- 이벤트 모니터 파일의 크기 또는 수가 CREATE EVENT MONITOR문에 있는 MAXFILES 또는 MAXFILESIZE에 대해 지정된 값을 초과하는 경우, 오류(SQLSTATE 54031)가 발생합니다.
- 이벤트 모니터의 목표 경로(CREATE EVENT MONITOR문에 지정되어 있음)가 이미 다른 이벤트 모니터에 의해 사용되고 있는 경우, 오류(SQLSTATE 51026)가 발생합니다.

주

- 이벤트 모니터를 활성화하면 관련 계수기가 재설정됩니다.

예

다음 예는 SMITHPAY라는 이벤트 모니터를 활성화합니다.

```
SET EVENT MONITOR SMITHPAY STATE = 1
```

SET INTEGRITY

SET INTEGRITY[105]문은 다음 중 하나를 수행하는 데 사용됩니다.

- 하나 이상의 테이블에 대한 무결성 점검을 해제합니다. 생성된 컬럼에 대한 값 생성, 점검 제한조건 및 참조 제한조건 점검, DATALINK 무결성 점검이 여기에 포함됩니다. 테이블이 REFRESH IMMEDIATE가 지정된 요약 테이블이면, 데이터의 즉시 새로 고침 기능은 중단됩니다. 테이블이 제한된 명령문 집합 및 명령에 의해서만 액세스될 수 있는 *점검 보류 상태*에 놓이게 됨을 유념하십시오. 기본 키 및 고유 제한조건은 계속해서 점검됩니다.
- 하나 이상의 테이블에 대한 무결성 점검을 다시 작동시키고, 지연된 모든 점검을 수행합니다. 테이블이 요약 테이블이면 필요에 따라 데이터가 화면갱신되며, REFRESH IMMEDIATE 속성이 정의되어 있으면 데이터의 즉시 화면 갱신 기능이 작동됩니다.
- 먼저 지연된 무결성 점검을 수행하지 않고 하나 이상의 테이블에 대한 무결성 점검을 설정합니다. 테이블이 REFRESH IMMEDIATE 속성이 정의된 요약 테이블이면, 데이터의 즉시 새로 고침 기능이 작동됩니다.
- 테이블이 이미 DRP(DataLink Reconcile Pending) 또는 DRNP(DataLink Reconcile Not Possible) 상태에 있는 경우 테이블을 점검 보류 상태로 둡니다. 테이블이 이러한 상태에 있지 않은 경우 무조건 테이블을 DRP 상태 및 점검 보류 상태로 설정하십시오.

로드된 후 테이블에 대한 무결성을 점검하기 위해 명령문이 사용될 때에, 시스템은 기본으로 제한조건 위반에 대한 첨부 부분만을 점검하여 테이블을 점증적으로 처리합니다. 그러나 시스템에서 전체 처리(제한조건 위반에 대해 테이블 전체를 점검함으로써)시 데이터 무결성을 확인해야 하는지를 결정하게 되는 몇 가지 상황이 있습니다. INCREMENTAL 옵션을 지정하여 사용자가 점증 처리를 명시적으로 요청해야 하는 상황도 있습니다. 1166 페이지의 『주』에서 자세한 내용을 참조하십시오.

SET INTEGRITY문은 트랜잭션 제어하에 있습니다.

105. SET CONSTRAINTS문보다는 SET INTEGRITY문이 DB2에서의 무결성 점검에 선호됩니다.

호출

이 명령문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다. 그러나, DYNAMICRULES BIND 바인드 옵션이 적용되면, 명령문은 동적으로 준비될 수 없습니다(SQLSTATE 42509).

권한 부여

SET INTEGRITY 실행에 필요한 특권은 아래에 대략적으로 나온 바와 같이 명령문 사용에 따라 달라집니다.

1. 무결성 점검을 끄십시오.

   명령문의 권한 부여 ID 특권에는 적어도 다음 중 하나가 포함됩니다.

   - 참조 무결성 제한조건에서 테이블 및 그의 모든 종속과 하위 테이블에 대한 CONTROL 특권
   - SYSADM 또는 DBADM 권한
   - LOAD 권한

2. 무결성 점검 및 점검 수행을 모두 켜십시오.

   명령문의 권한 부여 ID 특권에는 적어도 다음 중 하나가 포함됩니다.

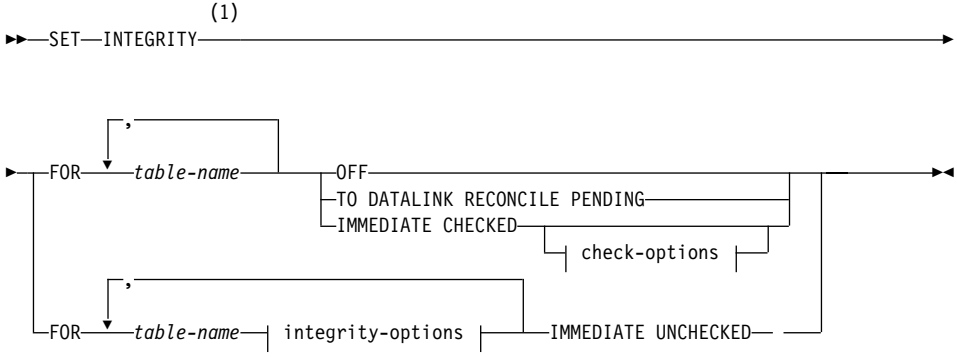
   - SYSADM 또는 DBADM 권한
   - 점검 중인 테이블에 대한 CONTROL 특권과 예외가 하나 이상의 테이블로 전달 중인 경우 예외 테이블에 대한 INSERT 특권
   - LOAD 권한 및 예외가 하나 이상의 테이블로 전달되는 경우
     - 점검 중인 각 테이블에 대한 SELECT 및 DELETE 특권
     - 예외 테이블에 대한 INSERT 특권.

3. 우선 점검을 수행하지 않고 무결성 점검을 작동시키는 경우,

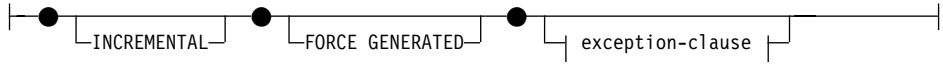
   명령문의 권한 부여 ID에는 최소한 다음 중 하나가 있어야 합니다.

   - SYSADM 또는 DBADM 권한
   - 점검 중인 테이블에 대한 CONTROL 특권
   - LOAD 권한

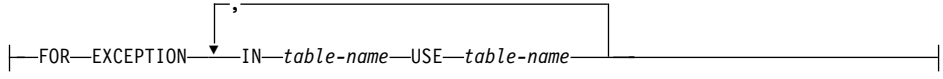
구문



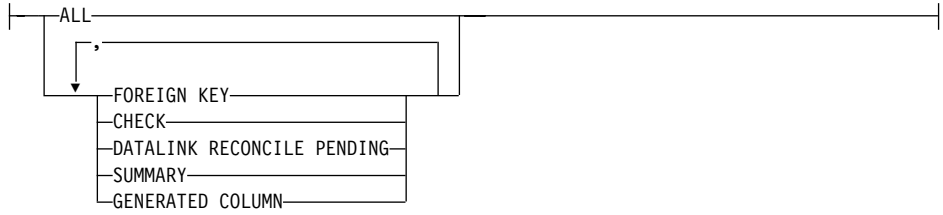
check-options:



exception-clause:



integrity-options:



주:

1 이전 버전과의 호환성을 위해, 키워드 CONSTRAINTS가 계속 지원될 것입니다.

설명

table-name

무결성 처리를 위해 테이블을 식별합니다. 카탈로그에 기술된 테이블이어야 하고 뷰, 카탈로그 테이블 또는 입력된 테이블이어서는 안 됩니다.

OFF

테이블의 외부 키 제한조건, 점검 제한조건 및 컬럼 생성이 해제되고 따라서 테이블이 점검 보류 상태에 놓이게 됨을 지정합니다. 요약 테이블일 경우, 즉시 새로 고침은 작동이 중지되며(적용가능한 경우) 요약 테이블은 점검 보류 상태에 놓입니다.

한 유형의 무결성 점검만 중지된 상태에서는 테이블이 이미 점검 보류 상태일 수 있음에 유의하십시오. 이 경우, 다른 유형의 무결성 점검도 작동중지됩니다.

목록에 있는 테이블이 상위 테이블인 경우, 외부 키 의무 규정에 대한 점검 보류 상태는 모든 종속 및 하위 테이블로 확장됩니다.

목록 내의 모든 테이블이 요약 테이블의 기본 테이블인 경우, 점검 보류 상태는 이러한 요약 테이블로까지 확대됩니다.

점검 보류 상태에 있는 테이블에서는 극히 한정된 활동만이 허용됩니다. 1166 페이지의 『주』는 의무 규정을 나열합니다.

TO DATALINK RECONCILE PENDING

DATALINK 무결성 제한조건 점검을 중지하고, 테이블이 점검 보류 상태에 있도록 지정합니다. 테이블이 이미 DRNP(DataLink Reconcile Not Possible) 상태에 있는 경우, 테이블은 점검 보류 상태로 남아 있습니다. 그렇지 않은 경우, 테이블이 DRP(DataLink Reconcile Pending) 상태로 설정됩니다.

이 옵션이 지정되는 경우 종속 및 하위 테이블은 영향받지 않습니다.

IMMEDIATE CHECKED

테이블의 무결성 점검이 실행되도록 하고 지연되었던 무결성 점검이 실행되도록 지정합니다. 이는 SYSCAT. TABLES 카탈로그의 STATUS_PENDING 및 CONST_CHECKED 컬럼에 있는 정보 세트에 따라 수행됩니다. 즉,

- STATUS의 값이 C(테이블이 점검 보류 상태에 있음)여야 하며, 아니면 오류(SQLSTATE 51027)가 발생합니다.

• CONST_CHECKED의 값은 어느 무결성 옵션이 점검될 지를 표시합니다.

요약 테이블인 경우, 데이터는 조회시 점검되며 필요에 따라 새로 고칩니다.

테이블이 DRP 또는 DRNP 상태에 있어도 DATALINK 값들은 점검되지 않습니다. RECONCILE 명령이나 API를 사용하여 DATALINK 값을 조정해야 합니다. 테이블은 점검 보류 상태에서 벗어나지만 계속해서 DRP 또는 DRNP 플래그가 설정됩니다. DATALINK 값 조정 시기가 연기되면서 테이블을 사용할 수 있게 됩니다.

점검 옵션

FORCE GENERATED

테이블에 생성된 컬럼이 있는 경우, 값은 표현식에 기초하여 계산되고 해당 컬럼에 저장됩니다. 이 절이 지정되지 않을 경우, 현재 값은 마치 동등한 점검 제한조건이 존재하는 것처럼 계산된 표현식 값과 비교됩니다.

INCREMENTAL

테이블의 첨부된 부분(있다면)상의 지연된 무결성 점검의 응용프로그램을 지정합니다. 요청이 만족될 수 없다면(즉, 시스템이 무결성 점검을 위해 전체 테이블을 점검해야 한다고 발견한 경우), 오류(SQLSTATE 55019)가 리턴될 것입니다. 속성이 지정되지 않으면, 시스템이 점증 처리가 가능한지의 여부를 판별할 것입니다. 그렇지 않으면, 전체 테이블이 점검될 것입니다. 시스템에서 증분식 처리보다는 전체 처리(무결성에 대해 테이블 전체 점검)를 선호하는 상황에 대해서는 참고를 참조하십시오. INCREMENTAL 옵션이 필요한 상황과 이 옵션을 지정할 수 없는 상황에 대해서도 참고를 참조하십시오.

테이블이 점검 보류 상태에 있지 않으면, 오류(SQLSTATE 55019)가 리턴됩니다.

exception-clause

FOR EXCEPTION

외부 키 의무 규정 또는 점검 제한조건을 위반한 행은 예외 테이블로 복사되고 원래의 테이블에서 삭제됨을 표시합니다. 이러한 사용자 정의 테이블에 대해 1499 페이지의 『부록N. 예외 테이블』에서 자세한 정보를 참조하십시오. 오류가 검출된 경우에도, 다시 의무 규정이 발

동되고 테이블은 점검 보류 상태에서 해제됩니다. 경고(SQLSTATE 01603)가 발행되어, 하나 이상의 행이 예외 테이블로 이동하였음을 나타냅니다.

FOR EXCEPTION절이 지정되어 있지 않고 의무 규정이 위반된 경우, 첫번째로 검출된 위반사항만이 사용자에게로 리턴됩니다 (SQLSTATE 23514). 테이블에 위반이 발생한 경우, 그 테이블은 명령문이 실행되기 전에 있었던 점검 보류 상태로 남게 됩니다. 이 절은 *table-name*이 요약 테이블일 경우에는 지정될 수 없습니다(SQLSTATE 42997).

IN *table-name*
의무 규정을 위반한 행을 복사할 테이블을 지정합니다. 각 테이블에 대해 하나의 예외 테이블이 존재해야 합니다.

USE *table-name*
오류 행이 복사될 예외 테이블을 지정합니다.

integrity-options
IMMEDIATE UNCHECKED로 설정된 무결성 옵션을 정의하는 데 사용됩니다.

ALL
모든 무결성 옵션이 작동될 것을 표시합니다.

FOREIGN KEY
외부 키 의무 규정이 작동될 것을 표시합니다.

CHECK
점검 제한조건이 작동될 것을 표시합니다.

DATALINK RECONCILE PENDING
DATALINK 무결성 제한조건이 작동될 것을 표시합니다.

SUMMARY
이는 REFRESH IMMEDIATE 속성을 가진 요약 테이블에 대해 즉시 새로 고침이 켜져야 함을 나타냅니다.

GENERATED COLUMN
생성된 컬럼이 작동될 것을 표시합니다.

IMMEDIATE UNCHECKED

다음 중 하나를 지정합니다.

- 테이블이 무결성 위반을 점검하지 않고 무결성 점검이 작동되게 하거나(따라서, 점검 보류 상태에서 벗어남), 요약 테이블에서 즉시 새로 고침이 작동되고 점검 보류 상태에서 나오게 됩니다.

  이는 주어진 테이블에 대해 ALL을 지정하거나 점검 제한조건만을 해제할 때 CHECK를 지정함으로써 또는 해당 테이블에 대해 외부 키 제한조건만을 해제할 때 FOREIGN KEY를 지정함으로써 또는 해당 테이블에 대해 DATALINK 무결성 제한조건만을 해제할 때 DATALINK RECONCILE PENDING을 지정함으로써 또는 해당 요약 테이블에 대해 요약 테이블 조회 점검만을 해제할 때 SUMMARY를 지정함으로써 또는 해당 테이블에 대해 컬럼 생성만을 해제할 때 GENERATED COLUMN을 지정함으로써 지정됩니다.

- 테이블의 무결성 점검 유형 중 하나가 설정되었으나, 이 테이블은 점검 보류 상태로 남게 됩니다.

  이는 주어진 테이블에 대해 이러한 유형의 제한조건이 해제될 때 CHECK, FOREIGN KEY, SUMMARY, GENERATED COLUMN 또는 DATALINK RECONCILE PENDING만을 지정함으로써 지정됩니다.

목록에 명시적으로 포함되어 있지 않은 테이블로 상태 변경이 확장되지 않습니다.

종속 테이블의 상위 테이블이 점검 보류 상태에 있는 경우, 종속 테이블의 외부 키 의무 규정은 점검을 생략하도록 표시될 수 없습니다.(점검 제한조건 점검은 생략될 수 있습니다.)

데이터 무결성에 관해서는 이 옵션을 사용하기 전에 고려되어야 합니다. 『주』에서 참조하십시오.

주

- 점검 보류 상태의 테이블에 대한 효과
  - 다음 중 한 상태에 있는 테이블상에서 SELECT, INSERT, UPDATE 또는 DELETE를 사용할 수 없습니다.
    - 점검 보류 상태에 있는 경우

- 또는 점검 보류 상태에 있는 다른 테이블을 액세스해야 하는 경우

  예를 들어, 점검 보류 상태에 있는 종속 테이블로 연쇄되는 상위 테이블에서 행의 DELETE는 허용되지 않습니다.

– 대부분의 경우, 테이블에 추가된 새로운 제한조건은 즉시 실시됩니다. 그러나 테이블이 점검 보류 상태라면, 테이블이 점검 보류 상태에서 빠져 나올 때까지 새 제한조건의 점검이 지연됩니다.

– CREATE INDEX문은 점검 보류 상태에 있는 테이블을 참조할 수 없습니다. 마찬가지로, 기본 키나 고유한 제한조건을 추가하는 ALTER TABLE은 점검 보류 상태에 있는 어떠한 테이블도 참조할 수 없습니다.

– 점검 보류 상태에 있는 테이블상에서 EXPORT, IMPORT, REORG, REORGCHK 등의 유틸리티를 조작할 수 없습니다. IMPORT 유틸리티는 항상 즉시 의무 규정을 점검한다는 점에서 LOAD 유틸리티와 다름에 유의하십시오.

– 점검 보류 상태에 있는 테이블에서도 UNLOAD, LOAD, BACKUP, RESTORE, ROLLFORWARD, UPDATE SATISTICS, RUNSTATS, LIST HISTORY, ROLLFORWARD 등의 유틸리티가 허용됩니다.

– ALTER TABLE, COMMENT ON, DROP TABLE, CREATE ALIAS, CREATE TRIGGER, CREATE VIEW, GRANT, REVOKE, SET INTEGRITY문은 점검 보류 상태에 있는 테이블을 참조할 수 있습니다.

– 점검 보류 상태에 있는 테이블에 종속된 패키지, 뷰 및 기타 오브젝트는 실행시 테이블을 액세스할 때 오류를 리턴시킵니다.

SET INTEGRITY문으로 위반 행을 제거하는 것은 삭제 이벤트가 아닙니다. 그러므로, 트리거는 SET INTEGRITY문으로 활성화되지 않습니다. 유사하게 FORCE GENERATED 옵션을 사용하여 생성된 컬럼을 갱신하면 트리거가 활성화되지 않습니다.

• 증분식 처리가 기본 동작이므로, 대부분의 경우에는 INCREMENTAL 옵션이 필요하지 않습니다. 그러나 다음 두 경우에는 필요합니다.

– IMMEDIATE UNCHECKED 옵션으로 이전에 점검 보류 상태에서 벗어난 테이블에 대해 강제로 점증 처리하기 위해 기본으로, 시스템은 모든 데이터의 뮤결성을 검증하기 위해 전체 처리를 선택합니다. 이러한 기본 동작은 새

로 추가된 부분만을 점검하기 위해 INCREMENTAL 옵션을 지정함으로써 대체할 수 있습니다. (세부사항은 "IMMEDIATE UNCHECKED절 사용에 대한 경고"를 참조하십시오.)

– 무결성 점검이 반드시 점증적으로 처리되게 하기 위해. INCREMENTAL 옵션을 지정하여, 시스템은 데이터 무결성을 보장하기 위해 전체 처리가 필요한 경우에 오류를 리턴합니다(SQLSTATE 55019).

• IMMEDIATE UNCHECKED절 사용에 관한 경고

– 이 절은 유틸리티 프로그램에 의해 사용되고, 응용프로그램에 의해 사용되는 것은 바람직하지 않습니다.

지연된 점검을 수행하지 않고 무결성 점검이 설정되었다는 사실이 카탈로그에 기록됩니다(SYSCAT.TABLES 뷰의 CONST_CHECKED 컬럼 값은 'U'로 설정됩니다). 이는 사용자가 특정 의무 규정에 대해서 데이터 무결성에 대한 책임이 있다는 것을 나타냅니다. 이 값은 다음 상황 중 하나가 될 때까지 남아 있습니다.

- 테이블은 CONST_CHECKED 컬럼의 'U' 값이 'W' 값으로 변경될 때 다시 점검 보류 상태가 되며(OFF절의 SET INTEGRITY문에서 테이블을 참조함으로써), 이는 사용자가 이전에 데이터 무결성을 책임진다고 했고 시스템이 데이터를 검증해야 함을 의미합니다.
- 테이블에 대해 점검되지 않은 모든 의무 규정은 삭제됩니다.
- REFRESH TABLE문은 요약 테이블에 대해 발행됩니다.

'W' 상태는 이전에 사용자가 무결성을 점검했고 아직 시스템에서는 무결성을 점검하지 않았으며 선택사항이 제공된 경우 시스템에서 데이터 무결성에 대해 테이블 전체를 다시 점검한 다음 이를 'Y' 상태로 변경한다는 사실을 기록한다는 점에서 'N' 상태와는 다릅니다. 아무 선택도 주어지지 않으면(즉, IMMEDIATE UNCHECKED 또는 INCREMENTAL이 지정될 때), 일부 데이터가 아직 시스템에 의해 검증되지 않았음을 기록하는 'U' 상태로 다시 변경됩니다. 후자(INCREMENTAL)의 경우에는, 경고(SQLSTATE 01636)가 리턴됩니다.

• Load Insert를 사용하여 데이터를 첨부한 후, SET INTEGRITY ... IMMEDIATE CHECKED문은 제한조건 위반에 대해 테이블을 점검한 루 테이블을 점검 보류 상태에서 빼냅니다. 시스템은 테이블에 대한 점증 처리가 가

능한지 여부를 결정합니다. 그런 경우에는, 첨부된 부분만이 무결성 위반에 대해 점검됩니다. 그렇지 않은 경우에는 시스템에서 무결성 위반에 대해 테이블 전체를 점검합니다(시스템이 전체 처리를 선호하는 상황에 대해서는 아래 참조).

- 다음은 사용자가 T IMMEDIATE CHECKED의 SET INTEGRITY문에 대해 INCREMENTAL 옵션을 지정하지 않은 경우 시스템에서 테이블 전체의 무결성을 점검하는 상황입니다.
  1. 테이블 T가 SYSCAT.TABLES 카탈로그의 CONST_CHECKED 컬럼에 하나 이상의 'W' 값을 가질 때.
- 다음은 시스템에서 T IMMEDIATE CHECKED의 SET INTEGRITY문에 대해 테이블 전체의 무결성을 점검하는 상황입니다(INCREMENTAL 옵션을 지정할 수 없습니다).
  1. 새 제한조건이 T 자체나 점검 보류 상태인 그의 상위 중 하나에 추가된 경우
  2. 로드 바꾸기가 T에 발생하거나 T에 대한 최종 무결성 점검 이후 NOT LOGGED INITIALLY WITH EMPTY TABLE 옵션이 활성화된 경우
  3. (전체 처리의 연쇄 효과) T의 상위가 무결성에 대해 비증분식으로 로드 바꾸기 또는 점검된 경우
  4. 이주 전에 테이블이 점검 보류 상태인 경우에는 이주 전에 처음으로 테이블 무결성이 점검될 때 전체 처리가 필요합니다.
  5. 테이블이나 그의 상위를 포함하는 테이블 공간이 특정 시점으로 롤 포워드된 경우
- DRNP(DataLink Reconcile Not Possible) 상태에 있는 테이블은 데이터베이스의 외부에서 조치가 취해질 필요가 있습니다. 일단 조치가 이루어지면 IMMEDIATE UNCHECKED 옵션을 사용하면서 DRNP 상태의 외부에서 테이블이 취해질 필요가 있습니다. RECONCILE 명령이나 API는 DATALINK 통합 제한조건을 점검하기 위해 사용되어야 합니다. *관리 안내서*에서 자세한 정보를 참조하십시오.
- 통합이 점검되는 동안, SET INTEGRITY에서 지정된 각각의 테이블에서 배타적 잠금이 취해집니다.
- 공유된 잠금은 SET INTEGRITY에 나열되지 않은 각각의 테이블로 알려지지만 점검되고 있는 의존 테이블 중 하나의 상위 테이블입니다.

- 통합 점검중 오류가 발생하면 원래의 것에서 지워지고 예외 테이블로 삽입되는 것을 포함해서 점검의 효과가 롤 백될 것입니다.
- FORCE GENERATED절과 함께 발행된 SET INTEGRITY문이 로그 공간 부족이나 로그 공간을 충분히 증가시킬 수 없어서 실패한 경우, 중간 확약을 사용하고 **db2gncol** 명령을 사용하여 값을 생성할 수 있습니다. 그런 다음 SET INTEGRITY를 FORCE GENERATED절없이 재수행할 수 있습니다.

예

예 1: 다음은 테이블의 점검 보류 상태에 대한 정보를 제공하는 조회의 예입니다. SUBSTR은 SYSCAT.TABLES의 CONST_CHECKED 컬럼에 있는 처음 2바이트를 추출하는 데 사용됩니다. 첫번째 바이트는 외부 키 의무 규정을 나타내고, 두번째 바이트는 점검 제한조건을 나타냅니다.

```
SELECT TABNAME,
  SUBSTR( CONST_CHECKED, 1, 1 ) AS FK_CHECKED,
  SUBSTR( CONST_CHECKED, 2, 1 ) AS CC_CHECKED
  FROM SYSCAT.TABLES
  WHERE STATUS = 'C'
```

예 2: 다음은 점검 보류 상태에 있는 T1 및 T2입니다.

```
SET INTEGRITY FOR T1, T2 OFF
```

예 3: T1에 대한 무결성을 점검하고 첫번째 위반만을 가져오십시오.

```
SET INTEGRITY FOR T1 IMMEDIATE CHECKED
```

예 4: T1 및 T2에 대한 무결성을 점검하고 위반 행을 예외 테이블 E1과 E2에 두십시오.

```
SET INTEGRITY FOR T1, T2 IMMEDIATE CHECKED
   FOR EXCEPTION IN T1 USE E1,
                 IN T2 USE E2
```

예 5: IMMEDIATE CHECKED 옵션으로 생략될 T1에서의 FOREIGN KEY 제한조건 점검을 사용하고 T2에서의 CHECK 제한조건 점검을 사용하십시오.

```
SET INTEGRITY FOR T1 FOREIGN KEY,
                  T2 CHECK IMMEDIATE UNCHECKED
```

예 6: 두 개의 ALTER TABLE문을 사용하여 점검 제한조건과 외부 키를 EMP_ACT 테이블에 추가하십시오. 테이블 단일 통과시 제한조건 점검을 수행하려면 ALTER문 이전에 무결성 점검이 작동 중지되고 실행 후에 점검됩니다.

```
SET INTEGRITY FOR EMP_ACT OFF;
ALTER TABLE EMP_ACT ADD CHECK (EMSTDATE <= EMENDATE);
ALTER TABLE EMP_ACT ADD FOREIGN KEY (EMPNO) REFERENCES EMPLOYEE;
SET INTEGRITY FOR EMP_ACT IMMEDIATE CHECKED
```

예 7: 다음은 생성된 컬럼에 대한 무결성 설정입니다.

```
SET INTEGRITY FOR T1 IMMEDIATE CHECKED
   FORCE GENERATED
```

SET PASSTHRU

SET PASSTHRU문은 데이터 소스의 원시 SQL을 그 데이터 소스에 직접 제출하기 위한 세션을 열고 닫습니다. 명령문은 트랜잭션 제어하에 있지 않습니다.

호출

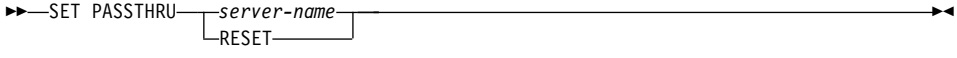
이 명령문은 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

명령문의 권한 부여 ID에 의해 보유된 특권은 다음을 위해 권한 부여를 제공해야 합니다.

- 데이터 소스를 통과하기 위해
- 데이터 소스에서 보안 측정을 만족시키기 위해

구문



설명

server-name

통과 세션이 열리는 데이터 소스의 이름. *server-name*은 카탈로그에 기술되어 있는 데이터 소스를 식별해야 합니다.

RESET

통과 세션을 닫습니다.

주

통과 사용에 대한 지침 및 제한사항에 대해서는 1415 페이지의 『통과 기능 처리』에서 참조하십시오.

예

예 1: 데이터 소스 BACKEND로의 통과 세션 시작

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
```

예 2: PREPARE문에 대해 통과 세션 시작

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL PREPARE STMT FROM :PASS_THRU;
EXEC SQL EXECUTE STMT;
```

예 3: 통과 세션 종료

```
strcpy (PASS_THRU_RESET," SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

예 4: PREPARE 및 EXECUTE문을 사용한 통과 세션 종료

```
strcpy (PASS_THRU_RESET," SET PASSTHRU RESET");
EXEC SQL PREPARE STMT FROM :PASS_THRU_RESET;
EXEC SQL EXECUTE STMT;
```

예 5: 데이터 소스를 통과하는 세션 열기, 이 데이터 소스에 있는 테이블에 대한 클러스터된 색인 작성 및 통과 세션 닫기

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
EXEC SQL PREPARE STMT                  pass-through mode
   FROM "CREATE UNIQUE
         CLUSTERED INDEX TABLE_INDEX
         ON USER2.TABLE                table is not an
         WITH IGNORE DUP KEY";          alias
EXEC SQL EXECUTE STMT;
STRCPY (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

SET PATH

SET PATH문은 CURRENT PATH 특수 레지스터의 값을 변경합니다. 이는 트랜잭션 제어하에 있지 않습니다.

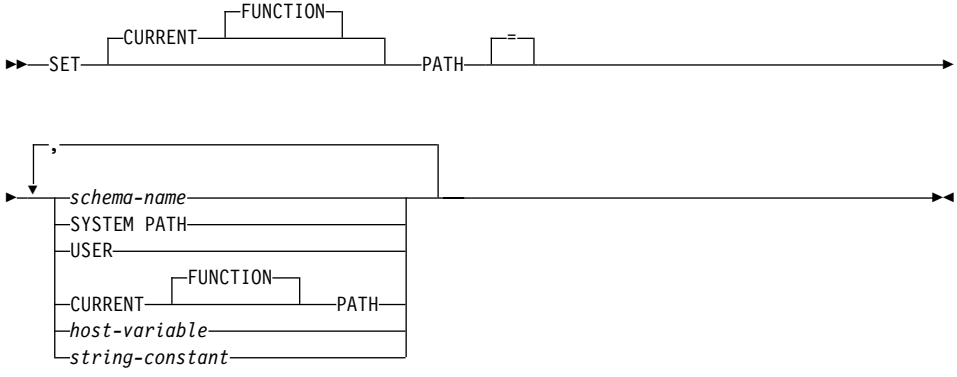
호출

이 명령문은 응용프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문을 실행하는 데 있어 권한은 필요없습니다.

구문



설명

스키마 이름

이 한 부분 설명은 응용프로그램 서버(AS)에 존재하는 스키마를 식별합니다. 경로 설정시에는 스키마가 존재하는지 확인되지 않습니다. 예를 들어, *스키마 이름* 철자가 잘못 입력된 경우, 검출되지는 않고 후속 SQL이 작동되는 방식에 영향을 줄 수 있습니다.

SYSTEM PATH

이 값은 스키마 이름 *"SYSIBM","SYSFUN"*을 지정하는 것과 같습니다.

USER

USER 특수 레지스터의 값.

CURRENT PATH

이 명령문을 실행하기 전 CURRENT PATH의 값. CURRENT FUNCTION PATH도 지정되어야 합니다.

host-variable

유형 CHAR 또는 VARCHAR의 변수. *host-variable* 내용의 길이는 30 바이트를 초과할 수 없습니다(SQLSTATE 42815). 이는 널(NULL)로 설정될 수 없습니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(NULL) 값을 표시해서는 안 됩니다(SQLSTATE 42815).

*host-variable*의 문자는 왼쪽 정렬되어야 합니다. *host-variable*로 *schema-name*을 지정할 때, 모든 문자는 대소문자를 정확히 구분하여 지정해야 합니다.(대문자로의 변환이 수행되지 않습니다.)

string-constant

최대 8자 길이의 문자열 상수

규칙

- 스키마 이름은 함수 경로에 한 번 이상 나타날 수 없습니다(SQLSTATE 42732).
- 지정될 수 있는 스키마 수는 CURRENT FUNCTION PATH 특수 레지스터의 총 길이로 제한됩니다. 특수 레지스터 문자열은 지정된 각 스키마 이름을 취하고 후미 공백을 제거하며, 큰 따옴표로 한계를 정하고, 필요한 스키마 이름 내에 큰 따옴표를 한 후 각 스키마 이름을 쉼표로 구분하여 만듭니다. 결과 문자열의 길이는 254바이트를 초과할 수 없습니다(SQLSTATE 42907).

주

- CURRENT PATH 특수 레지스터의 초기 값은 "SYSIBM","SYSFUN","X"입니다.(여기서, X는 USER 특수 레지스터의 값입니다.)
- 스키마 SYSIBM은 지정될 필요가 없습니다. SQL 경로에 포함되지 않으면, 이는 내재적으로 첫번째 스키마로 간주됩니다.(이 경우, CURRENT PATH 특수 레지스터에는 포함되지 않습니다.)

- CURRENT PATH 특수 레지스터는 동적 SQL문의 사용자 정의 데이터 유형, 프로시듀어 및 함수를 해석하는 데 사용되는 SQL 경로를 지정합니다. FUNCPATH 바인드 옵션은 정적 SQL문의 사용자 정의 데이터 유형 및 함수를 해석하는 데 사용되는 SQL 경로를 지정합니다. BIND 명령에서의 FUNCPATH 옵션 사용에 대한 자세한 정보는 *Command Reference*에서 참조하십시오.

예

예 1: 다음 명령문은 CURRENT FUNCTION PATH 특수 레지스터를 설정합니다.

```
SET PATH = FERMAT, "McDrw #8", SYSIBM
```

예 2: 다음 예는 CURRENT PATH 특수 레지스터의 현재 값을 CURPATH라는 호스트 변수로 읽어들입니다.

```
EXEC SQL VALUES (CURRENT PATH) INTO :CURPATH;
```

값은 이전 예에서 설정되는 경우, ″FERMAT″,″McDrw #8″,″SYSIBM″이 됩니다.

SET SCHEMA

SET SCHEMA문은 CURRENT SCHEMA 특수 레지스터의 값을 변경합니다. 이는 트랜잭션 제어하에 있지 않습니다. 패키지가 DYNAMICRULES BIND 옵션으로 바인드되는 경우 이 명령문은 아무 효과도 없습니다.

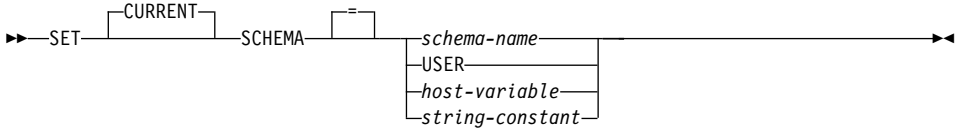
호출

명령문은 적용업무 프로그램에 포함되거나 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

이 명령문을 실행하는 데 있어 권한은 필요없습니다.

구문



설명

schema-name

이 한 부분 설명은 응용프로그램 서버(AS)에 존재하는 스키마를 식별합니다. 이름 길이는 30 바이트를 초과할 수 없습니다(SQLSTATE 42815). 그 스키마가 설정되는 시점에 스키마 존재 여부는 확인되지 않습니다. *schema-name* 철자를 잘못 입력할 경우, 검출되지는 않으나 후속 SQL이 작동하는 방식에 영향을 미칠 수 있습니다.

USER

USER 특수 레지스터의 값.

host-variable

유형 CHAR 또는 VARCHAR의 변수. *host-variable* 내용의 길이는 30 바이트를 초과할 수 없습니다(SQLSTATE 42815). 이는 널(NULL)로 설정될 수 없습니다. *host-variable*에 관련 표시기 변수가 있는 경우, 그 표시기 변수의 값은 널(NULL) 값을 표시해서는 안 됩니다(SQLSTATE 42815).

*host-variable*의 문자는 왼쪽 정렬되어야 합니다. *host-variable*로 *schema-name*을 지정할 때, 모든 문자는 대소문자를 정확히 구분하여 지정해야 합니다.(대문자로의 변환이 수행되지 않습니다.)

string-constant
최대 30자 길이의 문자열 상수

규칙

- 지정된 값이 *schema-name*에 대한 규칙을 지키지 않으면, 오류가 발생합니다 (SQLSTATE 3F000).
- CURRENT SCHEMA 특수 레지스터의 값은 모든 동적 SQL문에서 스키마로 사용되나, 데이터베이스 오브젝트에 대한 규정화되지 않은 참조가 존재하는 CREATE SCHEMA문은 예외입니다.
- QUALIFIER 바인드 옵션은 정적 SQL문에서 규정화되지 않은 데이터베이스 오브젝트 이름에 대한 규정자로 사용될 스키마 이름을 지정합니다.(QUALIFIER 옵션 사용에 대한 *Command Reference*에서 자세한 정보를 참조하십시오.)

주

- CURRENT SCHEMA 특수 레지스터의 초기 값은 USER 특수 레지스터의 초기 값과 동일합니다.
- CURRENT SCHEMA 특수 레지스터의 설정이 CURRENT PATH 특수 레지스터에 영향을 미치지는 않습니다. 따라서, CURRENT SCHEMA는 SQL 경로 및 함수에 포함되지 않으며, 프로시듀어 및 사용자 정의 유형 분석시 이러한 오브젝트를 찾지 못할 수 있습니다. SQL 경로에 현재 스키마 값을 포함시키려면, SET SCHEMA문을 발행할 때마다 SET SCHEMA문에 있는 스키마 이름을 포함하는 SET PATH문도 발행하십시오.
- CURRENT SQLID는 CURRENT SCHEMA의 동의어로 채택되며, SET CURRENT SQLID문이 미치는 영향은 SET CURRENT SCHEMA문이 미치는 영향과 동일합니다. 명령문 권한 부여 변경과 같은 다른 영향은 발생하지 않습니다.

예

예 1: 다음 명령문은 CURRENT SCHEMA 특수 레지스터를 설정합니다.

```
SET SCHEMA RICK
```

예 2: 다음 예는 CURRENT SCHEMA 특수 레지스터의 현재 값을 CURSCHEMA라는 호스트 변수로 읽어들입니다.

```
EXEC SQL VALUES (CURRENT SCHEMA) INTO :CURSCHEMA;
```

값은 이전 예에서 설정된 RICK입니다.

SET SERVER OPTION

SET SERVER OPTION문은 사용자나 응용프로그램이 연합 데이터베이스에 연결되어 있는 동안에도 계속 유효한 서버 옵션 설정값을 지정합니다. 연결이 종료될 때, 이 서버 옵션의 이전 설정값이 회복됩니다. 이 명령문은 트랜잭션 제어하에 있습니다.

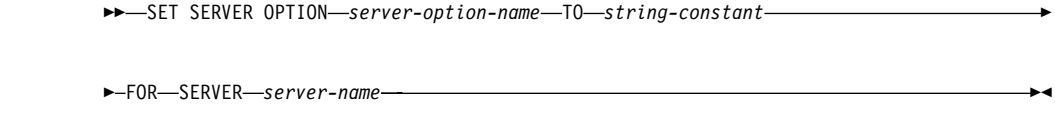
호출

이 명령문은 대화식으로 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

명령문의 권한 부여 ID는 연합 데이터베이스에서 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



설명

server-option-name

설정될 서버 옵션의 이름을 지정합니다. 서버 옵션의 1407 페이지의 『서버 옵션』에서 자세한 설명을 참조하십시오.

TO *string-constant*

*server-option-name*에 대한 설정값을 문자열 상수로서 지정합니다. 가능한 설정값에 대한 1407 페이지의 『서버 옵션』에서 자세한 설명을 참조하십시오.

SERVER *server-name*

*server-option-name*이 적용하는 데이터 소스의 이름을 지정합니다. 카탈로그에 기술된 서버여야 합니다.

주

- 서버 옵션 이름은 대문자나 소문자로 입력될 수 있습니다.
- SET SERVER OPTION은 현재 암호, fold_id 및 fold_pw 서버 옵션에만 지원합니다.
- 사용자나 응용프로그램이 연합 데이터베이스에 연결할 때 하나 이상의 SET SERVER OPTION문이 제출될 수 있습니다. 명령문은 작업 단위(UOW)의 시작에 지정되어야 하며 연결이 형성된 후에 처리되어야 합니다.

예

예 1: RATCHIT라는 Oracle 데이터 소스가 DJDB라는 연합 데이터베이스에 정의됩니다. RATCHIT는 플랜 힌트를 불허하도록 구성됩니다. 그러나, DBA는 새로운 응용프로그램의 시험 수행을 위해 플랜 힌트가 작동가능화되기를 원합니다. 수행이 끝나면, 플랜 힌트는 다시 불허될 것입니다.

```
CONNECT TO DJDB;
strcpy(stmt,"set server option plan_hints to 'Y' for server ratchit");
EXEC SQL EXECUTE IMMEDIATE :stmt;
strcpy(stmt,"select c1 from ora_t1 where c1 > 100"); /*Generate plan hints*/
EXEC SQL PREPARE s1 FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR s1;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :hv;
```

예 2: 모든 Oracle 8 데이터 소스에 대해 서버 옵션 PASSWORD를 'Y'(예는 데이터 소스에 있는 암호를 유효화함을 의미함)로 설정했습니다. 그러나 특정 Oracle 8 데이터 소스(연합 데이터베이스 DJDB를 ORA8A로 정의한)에 액세스하기 위해 응용프로그램이 연합 데이터베이스에 연결되어 있는 특정 세션의 경우에는 암호의 유효성을 확인할 필요가 없습니다.

```
CONNECT TO DJDB;
strcpy(stmt,"set server option password to 'N' for server ora8a");
EXEC SQL PREPARE STMT_NAME FROM :stmt;
EXEC SQL EXECUTE STMT_NAME FROM :stmt;
strcpy(stmt,"select max(c1) from ora8a_t1");
EXEC SQL PREPARE STMT_NAME FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR STMT_NAME;
EXEC SQL OPEN c1; /*Does not validate password at ora8a*/
EXEC SQL FETCH c1 INTO :hv;
```

SET 전이 변수

SET 전이 변수 명령문은 새로운 전이 변수에 값을 지정합니다. 이는 트랜잭션 제어하에 있습니다.

호출

이 명령문은 그 세분화도 FOR EACH ROW인 BEFORE 트리거 조치에서 트리거된 SQL문으로서만 사용됩니다(879 페이지의 『CREATE TRIGGER』 참조).

권한 부여

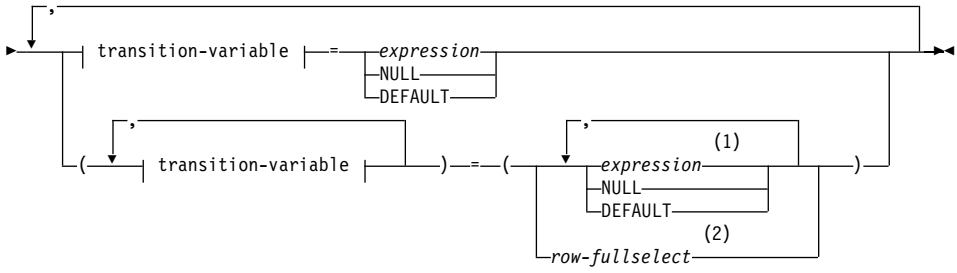
트리거 작성자의 권한 부여 ID에서 갖고 있는 특권에는 적어도 다음 사항 중 하나가 포함되어야 합니다.

- 할당된 왼쪽에서 참조되는 컬럼에는 UPDATE, 그리고 오른쪽에서 참조되는 컬럼에는 SELECT.
- 테이블에 대한 CONTROL 특권(트리거의 주제 테이블)
- SYSADM 또는 DBADM 권한

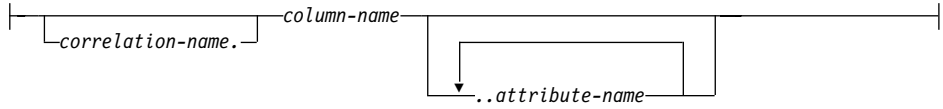
지정의 오른쪽에서 *row-fullselect*로 이 명령문을 실행하려면, 트리거 작성자의 권한 부여 ID에서 갖고 있는 특권에는 또한 참조되는 각 테이블 또는 뷰에 대해 적어도 다음 중 하나가 포함되어야 합니다.

- SELECT 특권
- CONTROL 특권
- SYSADM 또는 DBADM.

구문



transition-variable:



주:

1 표현식의 수, NULL 및 DEFAULT는 *transition-variable*의 수와 일치해야 합니다.

2 선택 목록의 컬럼 수는 *transition-variable*의 수와 일치해야 합니다.

설명

transition-variable

트리거에 대해 영향받는 행 세트에서 컬럼을 식별합니다.

correlation-name

NEW 전이 변수를 참조하기 위해 부여된 *correlation-name*. 이 *correlation-name*은 CREATE TRIGGER의 REFERENCING절에 있는 NEW 다음에 지정된 상관 이름과 일치해야 합니다.

REFERENCING절에 OLD가 지정되어 있지 않는 경우, *correlation-name*의 기본값은 NEW 다음에 지정된 *correlation-name*이 됩니다. NEW와 OLD가 모두 REFERENCING절에 지정되어 있는 경우, 각 *column-name*과 함께 *correlation-name*이 필요합니다(SQLSTATE 42702).

column-name

갱신될 컬럼을 식별합니다. *column-name*은 트리거의 주제 테이블 컬럼을 식별해야 합니다(SQLSTATE 42703). 컬럼은 한 번 이상 지정되어서는 안 됩니다(SQLSTATE 42701).

..attribute name

설정된 구조화 유형의 속성을 지정합니다(*attribute assignment*라고도 함). 지정된 *column-name*을 사용자 정의 구조화 유형으로 정의해야 합니다(SQLSTATE 428DP). *attribute-name*은 *column-name*의 구조화 유형 속성이어야 합니다(SQLSTATE 42703). *..attribute name* 절을 포함하지 않는 지정은 전통적인 지정이라고도 합니다.

expression

컬럼의 새 값을 나타냅니다. 이 표현식은 182 페이지의 『표현식』에 기술되어 있는 모든 유형의 표현식을 말합니다. 표현식에는 스칼라 fullselect 내에서 발생할 때를 제외하고 컬럼 함수를 포함시킬 수 없습니다(SQLSTATE 42903). *expression*에는 OLD 및 NEW 전이 변수 참조가 포함될 수 있고, 전이 변수를 지정하는 *correlation-name*으로 규정화해야 합니다(SQLSTATE 42702).

NULL

널(NULL) 값을 지정하고 널(NULL) 입력 가능 컬럼에 대해서만 지정될 수 있습니다(SQLSTATE 23502). 널(NULL)은 특별히 속성의 데이터 유형으로 변환되지 않을 경우 속성 지정의 값이 될 수 없습니다(SQLSTATE 429B9).

DEFAULT

해당 컬럼이 테이블에 정의된 방식에 기초하여 기본값을 사용하도록 지정됩니다. 삽입되는 값은 컬럼 정의 방식에 따릅니다.

- WITH DEFAULT절을 사용하여 컬럼이 정의된 경우, 값은 컬럼에 정의된 기본값으로 설정됩니다(524 페이지의 『ALTER TABLE』의 기본 절 참조).
- IDENTITY절을 사용하여 컬럼을 정의했다면 데이터베이스 관리 프로그램이 값을 생성합니다.
- WITH DEFAULT절, IDENTITY절 또는 NOT NULL절을 지정하지 않고 컬럼이 정의되었다면 값은 NULL입니다.

- NOT NULL절을 사용하여 컬럼을 정의했고 IDENTITY절이 사용되지 않았거나 또는 WITH DEFAULT절이 사용되지 않았거나 DEFAULT NULL이 사용되었다면 해당 컬럼에 대해 DEFAULT 키워드를 지정할 수 없습니다(SQLSTATE 23502).

row-fullselect

할당에 지정된 컬럼 이름의 수와 일치하는 컬럼수로 단일 행을 리턴하는 fullselect. 각 해당 컬럼 이름으로 값들이 할당됩니다. row-fullselect 결과 행이 없는 경우, 널(NULL) 값이 할당됩니다. *row-fullselect*에는 사용할 전이 변수를 지정하기 위해 *correlation-name*으로 규정화되어야 하는 OLD 및 NEW 전이 변수에 대한 참조가 포함됩니다(SQLSTATE 42702). 결과에 한 행 이상이 있는 경우 오류가 리턴됩니다(SQLSTATE 21000).

규칙

- 표현식으로부터 지정될 값, NULL 및 DEFAULT의 수 또는 *row-fullselect*는 할당 지정된 컬럼 수에 일치해야 합니다(SQLSTATE 42802).
- 명령문이 BEFORE UPDATE 트리거에 사용되는 경우, *transition-variable*로 지정되는 *column-name*은 파티션 키 컬럼이 될 수 없습니다(SQLSTATE 42997).

주

- 하나 이상의 할당이 포함된 경우, 모든 *expression* 및 *row-fullselect*는 할당이 수행되기 전에 평가됩니다. 따라서, 표현식의 컬럼 참조 또는 행 fullselect는 항상 단일 SET 전이 변수 명령문의 할당 이전의 전이 변수 값입니다.
- 구별 유형으로 정의된 식별 컬럼이 갱신된 경우, 전체 계산이 소스 유형으로 수행되고 그 값이 실제로 컬럼에 지정되기 전에 결과가 구별 유형으로 변환됩니다.[106]
- DB2로 하여금 SET문에서 식별 컬럼의 값을 생성하게 하려면 DEFAULT 키워드를 사용하십시오.

```
SET NEW.EMPNO = DEFAULT
```

106. 이전 값은 계산에 앞서 소스 유형으로 변환되지 않습니다.

이 예에서 NEW.EMPNO는 식별 컬럼으로 정의되고 이 컬럼을 갱신하는 데 사용되는 값은 DB2에 의해 생성됩니다.

- 식별 컬럼에 대해 생성된 순서 값 소모에 대해 자세히 알려면 1069 페이지의 『INSERT』를 참조하십시오.
- 식별 컬럼의 최대값 초과에 대해 자세히 알려면 1069 페이지의 『INSERT』를 참조하십시오.

예

예 1: 트리거 조치가 현재 실행 중인 행의 급여 컬럼을 50000으로 설정하십시오.

```
SET NEW_VAR.SALARY = 50000;
```

또는

```
SET (NEW_VAR.SALARY) = (50000);
```

예 2: 트리거 조치가 현재 실행 중인 행의 급여 및 상여금 컬럼을 각각 50000과 8000으로 설정하십시오.

```
SET NEW_VAR.SALARY = 50000, NEW_VAR.COMM = 8000;
```

또는

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (50000, 8000);
```

예 3: 트리거 조치가 현재 실행 중인 행의 급여 및 상여금 컬럼을 각각 갱신된 행 부서의 직원 급여 평균과 상여금 평균으로 설정하십시오.

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM)
   = (SELECT AVG(SALARY), AVG(COMM)
                    FROM EMPLOYEE E
                    WHERE E.WORKDEPT = NEW_VAR.WORKDEPT);
```

예 4: 트리거 조치가 현재 실행 중인 행의 급여 및 상여금 컬럼을 각각 10000과 급여의 원래 값으로 설정하십시오(SET문 실행 전에).

```
SET NEW_VAR.SALARY = 10000, NEW_VAR.COMM = NEW_VAR.SALARY;
```

또는

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (10000, NEW_VAR.SALARY);
```

SIGNAL SQLSTATE

SIGNAL SQLSTATE문은 오류를 신호하는 데 사용됩니다. 이로 인해, 지정된 SQLSTATE과 지정된 *진단 문자열*와 함께 오류가 리턴됩니다.

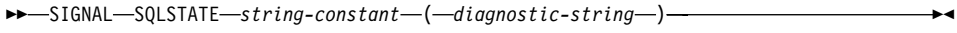
호출

SIGNAL SQLSTATE문은 트리거 내에서 트리거된 SQL문으로만 사용될 수 있습니다.

권한 부여

이 명령문을 실행하는 데 있어 권한은 필요없습니다.

구문



설명

문자열 상수

지정된 *리터럴*은 SQLSTATE를 나타냅니다. 이는 다음과 같은 응용프로그램 정의 SQLSTATE 규칙을 따르는 정확히 5자로 된 응용프로그램이어야 합니다.

- 각 문자는 숫자 집합('0' - '9')의 문자이거나, 강조되지 않은 대문자('A' - 'Z')여야 합니다.
- SQLSTATE 클래스(처음 두 자)는 오류 클래스가 아니므로 '00', '01' 또는 '02'가 될 수 없습니다.
- SQLSTATE 클래스(처음 두 자)가 문자 '0' - '6'이나 'A' - 'H'로 시작할 경우, 부속클래스(마지막 세 자)는 범위 'I' - 'Z' 범위 내의 문자로 시작해야 합니다.
- SQLSTATE 클래스(처음 두 자)가 문자 '7', '8', '9' 또는 'I' - 'Z'로 시작할 경우, 부속클래스(마지막 세 자)는 '0' - '9' 또는 'A' - 'Z'가 될 수 있습니다.

SQLSTATE가 이 규칙에 따르지 않으면 오류가 발생합니다(SQLSTATE 428B3).

진단 문자열

오류 조건을 기술하는 최고 70바이트의 문자열을 리턴하는 CHAR 또는 VARCHAR 유형을 가진 표현식. 문자열이 70바이트보다 크면, 그 문자열은 절단됩니다.

예

PARTS 테이블에 충분한 재고가 있는 경우에만 ORDERS 테이블(ORDERNO, CUSTNO, PARTNO, QUANTITY)의 주문을 기록하는 주문 시스템을 고려합니다.

```
CREATE TRIGGER check_avail
       NO CASCADE BEFORE INSERT ON orders
       REFERENCING NEW AS new_order
       FOR EACH ROW MODE DB2SQL
       WHEN (new_order.quantity > (SELECT on_hand FROM parts
                                   WHERE new_order.partno=parts.partno))
         BEGIN ATOMIC
           SIGNAL SQLSTATE '75001' ('Insufficient stock for order');
         END
```

UPDATE

UPDATE문은 테이블 또는 뷰 행의 지정된 컬럼 값을 갱신합니다. 뷰의 행을 갱신하면 그 기본 테이블의 행이 갱신됩니다.

이 명령문의 양식은 다음과 같습니다.

- *검색* UPDATE 양식은 하나 이상의 행을 갱신하는 데 사용됩니다(검색 조건에 의해 선택적으로 결정됨).
- *위치지정된* UPDATE 양식은 정확히 한 행을 갱신하는 데 사용됩니다(현재의 커서 위치에 의해 결정됨).

호출

UPDATE문은 응용프로그램에 포함되거나 동적 SQL문을 사용하여 발행될 수 있습니다. 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문입니다.

권한 부여

명령문의 권한 부여 ID에서 갖고 있는 특권은 다음 중 최소한 하나를 포함해야 합니다.

- 행이 갱신될 뷰 또는 테이블에서의 UPDATE 특권
- 갱신될 각 컬럼의 UPDATE 특권.
- 행이 갱신될 뷰 또는 테이블에서의 CONTROL 특권
- SYSADM 또는 DBADM 권한
- *행-fullselect*가 지정에 포함된 경우, 참조된 테이블 또는 뷰 각각에 대해 적어도 다음 중 하나가 있어야 합니다.
  - SELECT 특권
  - CONTROL 특권
  - SYSADM 또는 DBADM 권한

부속 조회에서 참조하는 각 테이블이나 뷰의 경우, 명령문의 권한 부여 ID가 보유한 특권에는 다음 중 적어도 하나는 포함되어야 합니다.

- SELECT 특권
- CONTROL 특권

- SYSADM 또는 DBADM 권한

패키지가 SQL92 규칙을 사용하여 사전 처리 컴파일되고 [107] UPDATE의 검색 형태에 *assignment-clause*의 오른쪽이나 *검색 조건*의 임의의 지점에 테이블이나 뷰의 컬럼에 대한 참조가 포함되는 경우, 명령문의 권한 부여 ID가 보유한 특권에는 다음 중 적어도 하나가 포함되어야 합니다.

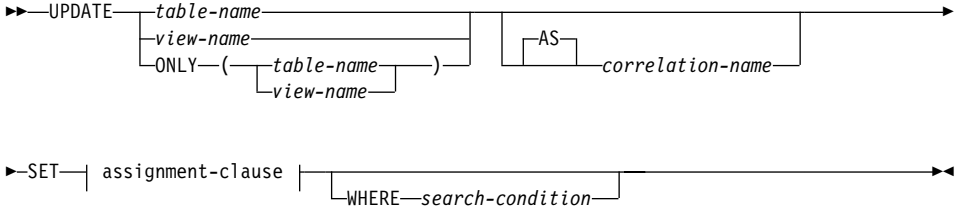
- SELECT 특권
- CONTROL 특권
- SYSADM 또는 DBADM 권한

지정된 테이블이나 뷰 앞에 ONLY 키워드가 올 때에는, 명령문의 권한 부여 ID에 의해 보유된 특권이 지정된 테이블이나 뷰의 모든 서브테이블이나 서브뷰에 대한 SELECT 특권도 포함해야 합니다.

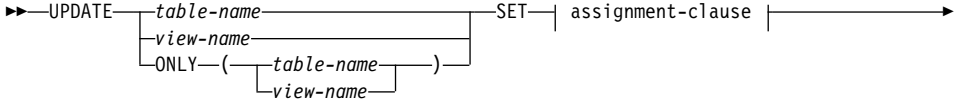
정적 UPDATE문에 대해서는 GROUP 특권이 점검되지 않습니다.

구문

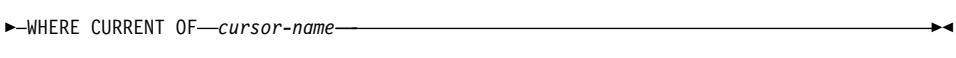
검색 UPDATE:



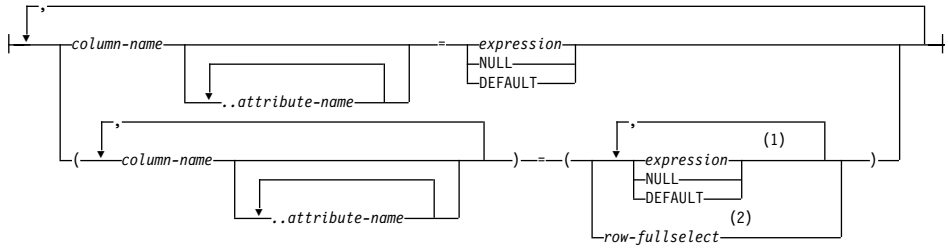
위치지정된 UPDATE:



107. 명령문을 처리하는 데 사용되는 패키지는 값이 SQL93E 또는 MIA인 LANGLEVEL 옵션을 사용하여 사전 처리 컴파일됩니다.



assignment-clause:



주:

1 표현식 NULL 및 DEFAULT의 수는 *column-name*의 수와 일치해야 합니다.

2 선택 목록의 컬럼 수는 *column-name* 수와 일치해야 합니다.

설명

table-name 또는 *view-name*

갱신될 뷰 또는 테이블 이름. 이름은 카탈로그에 있는 테이블 또는 뷰를 식별해야 하나 카탈로그 테이블, 카탈로그 테이블의 뷰(갱신 가능한 SYSSTAT 뷰 중 하나가 아니라면), 요약 테이블, 읽기 전용 뷰 또는 별명을 식별해서는 안됩니다. (읽기 전용 뷰 설명에 대해 931 페이지의 『CREATE VIEW』에서 참조하십시오. 갱신 가능 카탈로그 뷰에 대한 1281 페이지의 『부록D. 카탈로그 뷰』에서 참조하십시오.)

*테이블 이름*이 입력된 테이블인 경우, 테이블 행 또는 적절한 서브테이블이 명령문에 의해 갱신될 수 있습니다. 지정된 테이블의 컬럼들만이 WHERE절에서 설정되거나 참조될 수 있습니다. 위치 지정된 UPDATE의 경우, 연관된 커서가 ONLY를 사용하지 않고도 FROM절에서 동일한 테이블이나 뷰도 지정했을 것입니다.

ONLY (*table-name*)

입력된 뷰에 적용 가능한 ONLY 키워드는 명령문이, 지정된 테이블 데이터에만 적용되고 해당 서브테이블의 행은 갱신할 수 없도록 지정합니다. 위치지정

된 UPDATE인 경우, 연관된 커서가 ONLY를 사용하여 FROM절에 있는 테이블도 지정했을 것입니다. *table-name*이 입력된 테이블이 아닌 경우, ONLY 키워드는 명령문에 어떤 영향도 미치지 않습니다.

ONLY (*view-name*)

입력된 뷰에 적용 가능한 ONLY 키워드는 명령문이, 지정된 뷰 데이터에만 적용되고 해당 부속 뷰의 행은 갱신할 수 없도록 지정합니다. 위치지정된 UPDATE인 경우, 연관된 커서가 ONLY를 사용하여 FROM절에 있는 뷰도 지정했을 것입니다. *view-name*이 입력된 뷰가 아닌 경우, ONLY 키워드는 명령문에 어떤 영향도 미치지 않습니다.

AS

선택적 키워드로서 *correlation-name*을 도입합니다.

correlation-name

테이블 또는 뷰를 지정하기 위해 검색 조건 내에서 사용됩니다. (상관 이름 설명에 대해 149 페이지의 『상관 이름』에서 참조하십시오.)

SET

컬럼 이름에 대한 값 지정을 소개합니다.

assignment-clause

column-name

갱신될 컬럼을 식별합니다. *column-name*은 지정된 테이블이나 뷰의 갱신 가능 컬럼을 식별해야 합니다.[108] 입력된 테이블의 오브젝트 ID 컬럼은 갱신할 수 없습니다(SQLSTATE 428DZ). 컬럼은 속성 이름이 뒤에 오지 않을 경우 두 번 이상 지정해서는 안됩니다(SQLSTATE 42701).

위치지정된 UPDATE의 경우:

- UPDATE절이 커서의 select문에 지정되었으면, 지정 절의 각 컬럼 이름 또한 UPDATE절에 나타나야 합니다.

108. 파티션 키의 컬럼은 갱신할 수 없습니다(SQLSTATE 42997). 파티션 키의 컬럼을 변경하려면 데이터 행을 삭제 및 삽입해야 합니다.

- UPDATE절이 커서의 선택문에 지정되지 않았고 응용프로그램 사전 처리 컴파일시 LANGLEVEL MIA 또는 SQL92E가 지정되었다면, 갱신 가능 컬럼의 이름을 지정할 수 있습니다.
- UPDATE절이 커서의 선택문에 지정되지 않았고 응용프로그램 사전 처리 컴파일시 LANGLEVEL SAA1이 명시적으로 또는 기본값으로 지정되었다면, 컬럼을 갱신할 수 없습니다.

..attribute-name

설정된 구조화 유형의 속성을 지정합니다(*attribute assignment*라고도 함). 지정된 *column-name*을 사용자 정의 구조화 유형으로 정의해야 합니다(SQLSTATE 428DP). attribute-name은 *column-name*의 구조화 유형 속성이어야 합니다(SQLSTATE 42703). *..attribute-name* 절을 포함하지 않는 지정은 *전통적인 지정*이라고도 합니다.

expression

컬럼의 새 값을 나타냅니다. 이 표현식은 182 페이지의 『표현식』에 기술되어 있는 모든 유형의 표현식을 말합니다. 표현식에는 스칼라 fullselect 내에서 발생할 때를 제외하고 컬럼 함수를 포함시킬 수 없습니다(SQLSTATE 42903).

*expression*에는 UPDATE문의 목표 테이블의 컬럼 참조가 포함될 수 있습니다. 갱신되는 각 행에 대해 표현식에 있는 그러한 컬럼의 값은 행이 갱신되기 전에 행에 있는 컬럼값입니다.

NULL

널(NULL) 값을 지정하고 널(NULL) 입력 가능 컬럼에 대해서만 지정될 수 있습니다(SQLSTATE 23502). 널(NULL)은 특별히 속성의 데이터 유형으로 변환되지 않을 경우 속성 지정의 값이 될 수 없습니다(SQLSTATE 429B9).

DEFAULT

해당 컬럼이 테이블에 정의된 방식에 기초하여 기본값을 사용하도록 지정됩니다. 삽입되는 값은 컬럼 정의 방식에 따릅니다.

- 표현식을 기초로 해당 컬럼이 생성된 컬럼으로 정의된 경우, 시스템이 표현식을 근거로 컬럼 값을 생성합니다.

- IDENTITY절을 사용하여 컬럼을 정의했다면 데이터베이스 관리 프로그램이 값을 생성합니다.
- WITH DEFAULT절을 사용하여 컬럼이 정의된 경우, 값은 컬럼에 정의된 기본값으로 설정됩니다(524 페이지의 『ALTER TABLE』의 기본절 참조).
- WITH DEFAULT절, GENERATED절 또는 NOT NULL절을 지정하지 않고 컬럼이 정의되었다면 사용된 값은 NULL입니다.
- NOT NULL절을 사용하여 컬럼을 정의했고 GENERATED절이 사용되지 않았거나 또는 WITH DEFAULT절이 사용되지 않았거나 DEFAULT NULL이 사용되었다면 해당 컬럼에 대해 DEFAULT 키워드를 지정할 수 없습니다(SQLSTATE 23502).

GENERATED ALWAYS절로 정의된 생성된 컬럼을 설정할 수 있는 유일한 값은 DEFAULT입니다(SQLSTATE 428C9).

DEFAULT 키워드는 속성 지정의 값으로 사용될 수 없습니다(SQLSTATE 429B9).

row-fullselect

할당을 위해 지정된 *column-name*의 수와 일치하는 컬럼 수로 단일 행을 리턴하는 fullselect. 각각의 해당 *column-name*에 값이 할당됩니다. *row-fullselect* 결과에 행이 없는 경우, 널(NULL) 값이 할당됩니다.

*row-fullselect*에는 UPDATE문의 목표 테이블 컬럼 참조가 포함됩니다. 갱신되는 각 행에 대해 표현식에 있는 그러한 컬럼의 값은 행이 갱신되기 전에 행에 있는 컬럼값입니다. 결과에 한 행 이상이 있는 경우 오류가 리턴됩니다(SQLSTATE 21000).

WHERE

갱신되는 행을 표시하는 조건을 도입합니다. 절을 생략하거나 검색 조건을 제공하고 또는 커서를 명명할 수 있습니다. 절이 생략되는 경우, 테이블 또는 뷰의 모든 행이 갱신됩니다.

search-condition

73 페이지의 『제3장 언어 요소』에 기술된 검색 조건. 부속 조회가 아닌 검색 조건의 각 *column-name*은 테이블 또는 뷰의 컬럼을 명명해야 합니

다. 검색 조건에 UPDATE 및 부속 조회 모두의 기본 오브젝트가 같은 테이블인 부속 조회가 포함될 때, 부속 조회는 행이 갱신되기 전에 완전히 평가됩니다.

검색 조건은 테이블 또는 뷰의 각 행에 적용되며, 갱신된 행은 검색 조건 결과가 사실인 행입니다.

검색 조건에 부속 조회가 포함되는 경우, 부속 조회는 검색 조건이 한 행에 적용될 때마다 실행되고 그 결과는 검색 조건 적용시 사용된다고 생각할 수 있습니다. 실제로, 상관 참조가 없는 부속 조회는 한 번만 실행되는 반면, 상관 참조가 있는 부속 조회는 각 행에 대해 한 번 실행되어야 합니다.

CURRENT OF *cursor-name*

갱신 조작에 사용될 커서를 식별합니다. *cursor-name*은 952 페이지의 『DECLARE CURSOR』에 설명된 대로 선언된 커서를 식별해야 합니다. DECLARE CURSOR문은 프로그램에서 UPDATE문 앞에 와야 합니다.

명명된 테이블이나 뷰는 커서에 SELECT문의 FROM절에서도 명명되어야 하고, 커서의 결과 테이블은 읽기 전용이어야 합니다. (읽기 전용 테이블의 설명에 대해서는 952 페이지의 『DECLARE CURSOR』에서 참조하십시오.)

UPDATE문이 실행될 때 커서는 행에 위치해야 그 행이 갱신됩니다.

이러한 양식의 UPDATE는 갱신 목표가 뷰를 정의하는 fullselect의 선택 목록에 OLAP 함수를 포함하는 뷰인 경우 사용할 수 없습니다(SQLSTATE 42828).

규칙

- *지정*: 『제3장 언어 요소』에 기술된 지정 규칙에 따라 갱신 값이 컬럼에 지정됩니다.
- *유효성*: 갱신된 행은 갱신된 컬럼의 고유 색인에 의해 테이블(또는 뷰의 기본 테이블)에 강요되는 모든 제한조건을 따라야 합니다.

WITH CHECK OPTION을 사용하여 정의되지 않은 뷰가 사용된 경우, 뷰의 정의를 따르지 않도록 행이 변경될 수 있습니다. 그러한 행은 뷰의 기본 테이블에서 갱신되며, 뷰에는 더 이상 나타나지 않습니다.

WITH CHECK OPTION을 사용하여 정의된 뷰를 사용하는 경우, 갱신된 행은 그 뷰의 정의를 따라야 합니다. 이러한 상황에 적용되는 규칙에 대한 설명을 보려면 931 페이지의 『CREATE VIEW』에서 참조하십시오.

- *점검 제한조건*: 갱신 값은 테이블에 정의된 점검 제한조건의 점검 조건을 만족시켜야 합니다.

  정의된 점검 제한조건을 지닌 테이블로 UPDATE함으로써 갱신된 각 행에 대해 한 번 평가되고, 갱신된 각 컬럼에 대한 점검 제한조건을 가집니다. UPDATE 문 처리시, 갱신된 컬럼을 참조하는 점검 제한조건만이 점검됩니다.

- *참조 무결성*: 상위 고유 키의 값은 갱신 규칙이 제한되고 하나 이상의 종속 행이 있는 경우 변경될 수 없습니다. 그러나 갱신 규칙이 NO ACTION인 경우 갱신 명령문 완료시까지 모든 하위 키가 상위 키를 갖게 된다면, 상위 고유 키를 갱신할 수 있습니다. 외부 키의 NULL이 아닌 갱신 값은 관계의 상위 테이블에서 1차 키 값과 같아야 합니다.

주

- 갱신 값이 의무 규정을 위반하거나 UPDATE문 실행중에 오류가 발생하면, 행이 갱신되지 않습니다. 복수의 행이 갱신되는 순서는 정의되어 있지 않습니다.
- UPDATE문이 실행을 마칠 때 SQLCA의 SQLERRD(3) 값이 갱신된 행의 수입니다. SQLERRD(5) 필드에는 모든 활성 트리거가 삽입, 삭제 또는 갱신한 행 수가 들어 있습니다. SQLCA 설명에 관해서는 1261 페이지의 『부록B. SQL 통신(SQLCA)』에서 참조하십시오.
- 적절하게 잠금이 되어 있지 않는 한, 성공적인 UPDATE문의 실행시 하나 이상의 독점적인 잠금이 생깁니다. 잠금이 해제될 때까지 갱신된 행은 갱신을 수행한 응용프로그램 프로세스에 의해서만 액세스됩니다(미확약 읽기 분리 레벨을 사용하는 응용프로그램인 경우는 제외). 잠금에 대해 자세히 알려면, COMMIT, ROLLBACK 및 LOCK TABLE문 설명을 보십시오.
- DATALINK 컬럼의 URL 값이 갱신되는 경우, 이는 기존의 DATALINK 값을 삭제하고 새 값을 삽입하는 것과 같습니다. 우선, 기존의 값이 파일에 연결

된 경우 그 파일의 링크가 해제됩니다. 그러면, DATALINK 값의 링크 속성이 비어 있지 않는 한, 지정된 파일은 그 파일에 링크됩니다.

URL 경로로서 빈 문자열을 지정함으로써 파일을 다시 링크하지 않고 (예를 들어, DLVALUE 스칼라 함수의 *데이터 위치* 인수로서 또는 새로운 값이 기존 값과 같도록 지정함으로써) DATALINK 컬럼의 주석 값을 갱신할 수 있습니다.

널(NULL) 값으로 DATALINK 컬럼이 갱신되는 경우, 이것은 기존의 DATALINK 값을 삭제하는 것과 같습니다.

기존 값 또는 새 값의 파일 서버가 더 이상 데이터베이스 서버로 등록되지 않는 경우 DATALINK 값을 갱신하려 할 때 오류가 발생할 수 있습니다 (SQLSTATE 55022).

- 입력된 테이블에 대한 컬럼 분배 통계를 갱신할 때, 처음 컬럼을 도입한 서브테이블을 지정해야 합니다.
- 같은 구조화 유형에 대해 여러 속성 지정이 SET절에 지정된 순서대로 그리고 괄호로 묶인 SET절내에서 왼쪽으로부터 오른쪽순으로 발생합니다.
- 속성 지정은 사용자 정의 구조화 유형의 속성에 대한 변환 메소드를 호출합니다. 예를 들어, 지정 `st..a1=x`는 지정 `st = st..a1(x)`에서 변환 메소드를 사용할 때와 같은 효과를 지닙니다.
- 주어진 컬럼이 단 하나의 전통적인 지정에 있는 목표 컬럼일 수 있는 반면, 컬럼은 여러 속성 지정의 목표 컬럼일 수 있습니다(그러나 전통적인 지정의 목표 컬럼이 아닌 경우에만).
- 구별 유형으로 정의된 식별 컬럼이 갱신된 경우, 전체 계산이 소스 유형으로 수행되고 그 값이 실제로 컬럼에 지정되기 전에 결과가 구별 유형으로 변환됩니다.[109]
- DB2로 하여금 SET문에서 식별 컬럼의 값을 생성하게 하려면 DEFAULT 키워드를 사용하십시오.

```
SET NEW.EMPNO = DEFAULT
```

109. 이전 값은 계산에 앞서 소스 유형으로 변환되지 않습니다.

이 예에서 NEW.EMPNO는 식별 컬럼으로 정의되고 이 컬럼을 갱신하는 데 사용되는 값은 DB2에 의해 생성됩니다.

- 식별 컬럼에 대해 생성된 순서 값 소모에 대해 자세히 알려면 1069 페이지의 『INSERT』를 참조하십시오.
- 식별 컬럼의 최대값 초과에 대해 자세히 알려면 1069 페이지의 『INSERT』를 참조하십시오.

예

- *예 1:* EMPLOYEE 테이블의 직원 번호(EMPNO) '000290'의 작업(JOB)을 'LABORER'로 변경하십시오.

```
UPDATE EMPLOYEE
 SET JOB = 'LABORER'
 WHERE EMPNO = '000290'
```

- *예 2:* 부서(DEPTNO) 'D21'에서 담당하는 PROJECT 테이블의 모든 프로젝트에 대한 프로젝트 스탭핑(PRSTAFF)을 1.5만큼 증가십시오.

```
UPDATE PROJECT
 SET PRSTAFF = PRSTAFF + 1.5
 WHERE DEPTNO = 'D21'
```

- *예 3:* 부서(WORKDEPT) 'E21'의 관리자를 제외한 모든 직원이 임시로 재지정되었습니다. 이들의 직책(JOB)을 NULL로 변경하고 EMPLOYEE 테이블에서 급여(SALARY, BONUS, COMM) 값을 제로로 변경함으로써 이를 표시합니다.

```
UPDATE EMPLOYEE
 SET JOB=NULL, SALARY=0, BONUS=0, COMM=0
 WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

이 명령문 또한 다음과 같이 기록될 수 있습니다.

```
UPDATE EMPLOYEE
 SET (JOB, SALARY, BONUS, COMM) = (NULL, 0, 0, 0)
 WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

- *예 4:* 직원 번호 000120인 직원의 급여 및 상여금 컬럼을 각각 갱신된 행 부서의 직원 급여 및 상여금 평균으로 갱신하십시오.

```
UPDATE EMPLOYEE EU
 SET (EU.SALARY, EU.COMM)
 =
```

```
(SELECT AVG(ES.SALARY), AVG(ES.COMM)
   FROM EMPLOYEE ES
   WHERE ES.WORKDEPT = EU.WORKDEPT)
   WHERE EU.EMPNO = '000120'
```

- *예 5:* C 프로그램에서 EMPLOYEE 테이블의 행을 표시한 다음, 요청이 있을 경우 특정 직원의 작업(JOB)을 입력된 새 작업으로 변경하십시오.

```
EXEC SQL  DECLARE C1 CURSOR FOR
                SELECT *
                             FROM EMPLOYEE
                FOR UPDATE OF JOB;
EXEC SQL  OPEN C1;
EXEC SQL  FETCH C1 INTO ...     ;
if ( strcmp (change, "YES") == 0 )
  EXEC SQL  UPDATE EMPLOYEE
              SET JOB = :newjob
              WHERE CURRENT OF C1;
EXEC SQL  CLOSE C1;
```

- *예 6:* 이러한 예는 컬럼 오브젝트의 속성을 변환합니다.

  다음과 같은 유형과 테이블이 있다고 가정합니다.

```
CREATE TYPE POINT AS (X INTEGER, Y INTEGER)
  NOT FINAL WITHOUT COMPARISONS
  MODE DB2SQL

CREATE TYPE CIRCLE AS (RADIUS INTEGER, CENTER POINT)
  NOT FINAL WITHOUT COMPARISONS
  MODE DB2SQL

CREATE TABLE CIRCLES (ID INTEGER, OWNER VARCHAR(50), C CIRCLE
```

  다음 예는 ID가 999인 CIRCLE 컬럼의 RADIUS 속성과 OWNER 컬럼을 변경하여 CIRCLES 테이블을 갱신합니다.

```
 UPDATE CIRCLES
  SET OWNER = 'Bruce'
    C..RADIUS = 5
  WHERE ID = 999
```

  다음 예는 999로 식별되는 원의 중심에 대한 X 및 Y 좌표를 바꿉니다.

```
UPDATE CIRCLES
  SET C..CENTER..X = C..CENTER..Y,
    C..CENTER..Y = C..CENTER..X
  WHERE ID = 999
```

다음 예는 위의 두 명령문을 작성하는 또 다른 방법입니다. 이 예는 상위 두 예의 효과를 결합합니다.

```
UPDATE CIRCLES
 SET (OWNER,C..RADIUS,C..CENTER..X,C..CENTER..Y) =
   ('Bruce',5,C..CENTER..Y,C..CENTER..X)
 WHERE ID = 999
```

VALUES

VALUES문은 조회의 한 양식입니다. 이는 응용프로그램에 포함되거나, 대화식으로 발행될 수 있습니다. 476 페이지의 『fullselect』에서 자세한 정보를 참조하십시오.

VALUES INTO

VALUES INTO문은 많아야 한 행으로 구성되는 결과 테이블을 작성한 후, 그 행의 값을 호스트 변수에 지정합니다.

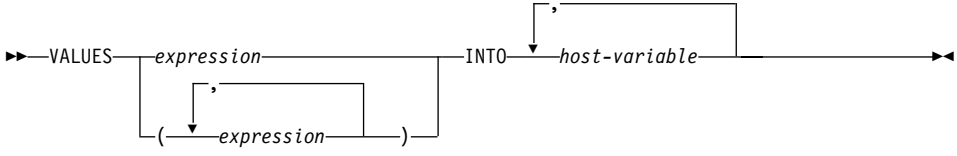
호출

이 명령문은 응용프로그램에만 포함될 수 있습니다. 이 명령문은 동적으로 준비될 수 없는 실행 가능 명령문입니다.

권한 부여

필요 사항 없음.

구문



설명

VALUES

하나 이상의 컬럼으로 구성되는 단일 행을 도입합니다.

표현식

단일 컬럼 결과 테이블의 단일 값을 정의하는 표현식.

(표현식,...)

결과 테이블의 하나 이상 컬럼에 대해 값을 정의하는 복수의 표현식.

INTO

호스트 변수 목록을 나열합니다.

호스트 변수

호스트 변수 선언 규칙에 따른 프로그램에 기술된 변수를 식별합니다.

결과 행의 첫번째 값은 목록의 첫번째 변수에 지정되고, 두 번째 값은 두 번째 호스트 변수에, 이와 같은 식으로 지정됩니다. 호스트 변수의 수가 컬

럼 값보다 작으면 'W' 값이 SQLCA의 SQLWARN3 필드에 지정됩니다 (1261 페이지의 『부록B. SQL 통신(SQLCA)』 참조).

109 페이지의 『지정 및 비교』에 설명되어 있는 규칙에 따라 변수가 각각 지정됩니다. 목록을 통해 순서대로 지정됩니다.

오류가 발생하면, 값이 호스트 변수에 지정되지 않습니다.

예

예 1: 이 C 예는 CURRENT PATH 특수 레지스터의 값을 호스트 변수로 읽어 들입니다.

```
EXEC SQL VALUES(CURRENT PATH)
         INTO :hv1;
```

예 2: 이 C 예는 LOB 필드의 부분을 호스트 변수로 검색하여, 지연된 검색에 대한 LOB 위치 지정자를 실행합니다.

```
EXEC SQL VALUES (substr(:locator1,35))
         INTO :details;
```

WHENEVER

WHENEVER문은 지정된 예외 조건이 발생할 때 취해야 할 조치를 지정합니다.

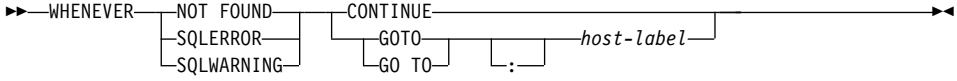
호출

이 명령문은 응용프로그램에 포함될 수 있습니다. 이 명령문은 실행되지 않습니다. 이 명령문은 REXX에서 지원되지 않습니다.

권한 부여

필요 사항 없음.

구문



설명

NOT FOUND, SQLERROR 또는 SQLWARNING절은 예외 조건 유형을 식별하는 데 사용됩니다.

NOT FOUND
SQLCODE가 +100 또는 SQLSTATE가 '02000'이 되는 조건을 식별합니다.

SQLERROR
음수 SQLCODE가 되는 조건을 식별합니다.

SQLWARNING
경고 조건(SQLWARN0는 'W')이 되거나, +100이 아닌 양수 SQL 리턴 코드가 되는 조건을 식별합니다.

식별된 예외 조건 유형이 존재할 때 수행할 일을 지정하는 데 CONTINUE 또는 GO TO절을 사용합니다.

CONTINUE
원시 프로그램의 다음 명령을 실행하도록 합니다.

GOTO 또한 **GO TO** *호스트 레이블*

호스트 레이블에 의해 식별되는 명령문으로 제어가 전달되도록 합니다. 호스트 레이블의 경우, 단일 토큰을 교체합니다. 이 때, 앞에 콜론을 둘 수 있습니다. 토큰의 형태는 호스트 언어에 따라 다릅니다.

주

세가지 유형의 WHENEVER문이 있습니다.

- WHENEVER NOT FOUND
- WHENEVER SQLERROR
- WHENEVER SQLWARNING

프로그램의 모든 실행 가능 SQL문은 각 유형의 암시적 또는 명시적 WHENEVER 문의 영역 내에 있습니다. WHENEVER문의 영역은 프로그램 내의 명령문 실행 순서가 아닌, 나열 순서와 관계 있습니다.

SQL문은 원시 프로그램의 SQL문 앞에 지정된 각 유형의 마지막 WHENEVER 문 영역 내에 있습니다. 일부 유형의 WHENEVER문이 SQL문 앞에 지정되지 않으면, 그 SQL문은 CONTINUE가 지정된 유형의 암시적 WHENEVER문 내에 있게 됩니다.

예

다음 C 예에서는, 오류가 발생하면 HANDLERR로 갑니다. 경고 코드가 생성되면, 정상적으로 진행되는 프로그램을 계속합니다. 데이터가 리턴되지 않으면, ENDDATA로 갑니다.

```
EXEC SQL WHENEVER SQLERROR GOTO HANDLERR;
EXEC SQL WHENEVER SQLWARNING CONTINUE;
EXEC SQL WHENEVER NOT FOUND GO TO ENDDATA;
```

제7장 SQL 프로시듀어

SQL 프로시듀어는 프로시듀어 본문에 있는 CREATE PROCEDURE문으로 구성되어 있습니다.

이 장에는 SQL 프로시듀어 명령문의 프로시듀어 본문에 대한 구문 및 매개변수 설명이 나와 있습니다.

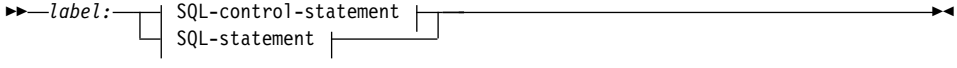
SQL 프로시듀어 명령문

SQL 저장 프로시듀어 정의의 프로시듀어 본문에는 저장 프로시듀어에 대한 소스 명령문이 포함됩니다.

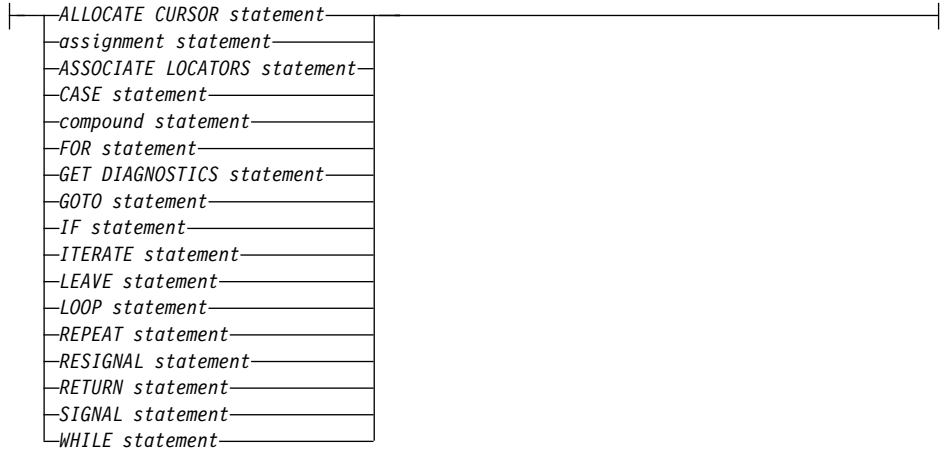
이 장에는 구문 다이어그램, 의미론적 설명, 규칙 및 프로시듀어 본문을 구성하는 명령문의 사용 예가 나와 있습니다.

SQL 제어 명령문이 SQL 프로시듀어 본문으로 지정된 경우, 해당 제어 명령문에 여러 명령문을 지정할 수 있습니다. 이러한 명령문은 SQL 프로시듀어 명령문으로 정의됩니다.

구문



SQL-control-statement:



설명

label:

SQL 프로시듀어 명령문에 대해 레이블을 지정합니다. 레이블은 SQL 프로시듀어 명령문 목록내에서 고유해야 합니다(이 목록에 중첩된 복합 텍스트 명령

문 포함). 중첩되지 않은 복합 텍스트 명령문은 같은 레이블을 사용할 수도 있음에 유의하십시오. SQL 프로시듀어 목록은 수많은 SQL 제어 명령문에서 가능합니다.

SQL문

모든 실행 가능 SQL문을 SQL 프로시듀어 본문에 포함시킬 수 있으나, 다음은 예외입니다.

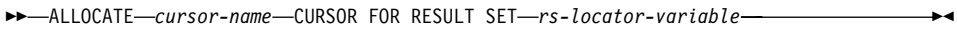
- CONNECT
- 색인, 테이블 또는 뷰가 아닌 모든 오브젝트 CREATE
- DESCRIBE
- DISCONNECT
- 색인, 테이블 또는 뷰가 아닌 모든 오브젝트 DROP
- FLUSH EVENT MONITOR
- REFRESH TABLE
- RELEASE(연결에서만)
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- SET CONNECTION
- SET INTEGRITY

주: SQL 프로시듀어 본문내에 CALL문을 포함시킬 수 있으나, 이러한 CALL 문만이 또 다른 SQL 프로시듀어를 호출할 수 있습니다. SQL 프로시듀어 본문내의 CALL문은 다른 유형의 저장 프로시듀어를 호출할 수 없습니다.

ALLOCATE CURSOR문

ALLOCATE CURSOR 문은 결과 세트 위치 지정자 변수로 식별되는 결과 세트에 대한 커서를 할당합니다. 1214 페이지의 『ASSOCIATE LOCATORS문』에 결과 세트 위치 지정자 변수에 대해 자세히 나와 있습니다.

구문



설명

cursor-name

커서 이름을 지정합니다. 이 이름은 이미 소스 SQL 프로시듀어에서 선언된 커서를 식별해야 합니다(SQLSTATE 24502).

CURSOR FOR RESULT SET *rs-locator-variable*

호스트 변수 규칙에 따라 소스 SQL 프로시듀어에 선언된 결과 세트 위치 지정자 변수를 지정합니다. SQL 변수 선언에 대해 자세히 알려면 1221 페이지의 『SQL variable declaration』을 참조하십시오.

결과 세트 위치 지정자 변수에는 ASSOCIATE LOCATORS SQL 문이 리턴하는 유효한 결과 세트 위치 지정자 값이 포함되어야 합니다(SQLSTATE 24501).

주

- **동적으로 준비된 ALLOCATE CURSOR 문**: USING 절을 사용한 EXECUTE 문을 사용하여 동적으로 준비된 ALLOCATE CURSOR 문을 실행해야 합니다. 동적으로 준비된 명령문에서 호스트 변수로의 참조는 매개변수 표시문자(물음표)로 표시됩니다.

  ALLOCATE CURSOR 문에서 *rs-locator-variable*은 항상 호스트 변수입니다. 따라서 동적으로 준비된 ALLOCATE CURSOR 문의 경우, EXECUTE 문의 USING 절은 매개변수 표시문자에 대해 그 값이 *rs-locator-variable*로 대체되는 호스트 변수를 식별해야 합니다.

- 이미 DECLARE CURSOR 문에 사용된 명령문 식별자로 ALLOCATE CURSOR 문을 준비할 수 없습니다. 예를 들어, 다음 SQL문은 유효하지 않은데, 이는 PREPARE 문이 STMT1을 ALLOCATE CURSOR 문의 식별자로 사용하고 STMT1이 이미 DECLARE CURSOR 문에 사용되었기 때문입니다.

```
DECLARE CURSOR C1 FOR STMT1;
PREPARE STMT1 FROM
   'ALLOCATE C2 CURSOR FOR RESULT SET ?';
```

규칙

- 다음 규칙은 할당된 커서를 사용할 때 적용됩니다.
  - 할당된 커서는 OPEN 문으로 열 수 없습니다(SQLSTATE 24502).
  - 할당된 커서는 CLOSE 문으로 닫을 수 있습니다. 할당된 커서를 닫으면 저장 프로시듀어의 관련 커서가 닫힙니다.
  - 각 결과 세트에 대해 하나의 커서만을 할당할 수 없습니다.
- 할당된 커서는 구간 복원 조작, 내재된 닫기 또는 명시적 닫기까지 지속됩니다.
- 확약 조작은 저장 프로시듀어에 의해 정의된 WITH HOLD 상태가 아닌 할당된 커서를 파괴합니다.
- 할당된 커서를 없애면 SQL 프로시듀어의 관련 커서가 닫힙니다.

예

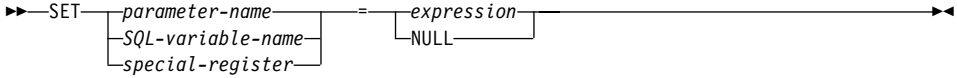
이 SQL 프로시듀어 예는 커서 C1을 정의하고 결과 세트 위치 지정자 변수 LOC1 및 SQL 프로시듀어에 의해 리턴된 관련 결과 세트와 연관시킵니다.

```
ALLOCATE C1 CURSOR FOR RESULT SET LOC1;
```

지정문

이 지정 명령문은 출력 매개변수, 지역 변수 또는 특수 레지스터에 값을 지정합니다.

구문



설명

parameter-name

지정 목표인 매개변수를 식별합니다. 매개변수를 CREATE PROCEDURE 문의 *parameter-declaration*에 지정해야 하고 OUT 또는 INOUT 매개변수로 정의해야 합니다.

SQL-variable-name

지정 목표인 SQL 변수를 식별합니다. SQL 변수는 사용하기 전에 선언해야 합니다. SQL 변수는 복합 텍스트 명령문에 정의할 수 있습니다.

special-register

지정 목표인 특수 레지스터를 식별합니다. CURRENT FUNCTION 또는 CURRENT SCHEMA 특수 레지스터를 포함한 특수 레지스터가 스키마 이름을 값으로 수용하는 경우, DB2는 지정 매개변수가 SQL 변수인지 여부를 판별합니다. 지정 매개변수가 SQL 변수인 경우, DB2는 SQL 변수의 값을 특수 레지스터에 지정합니다. 지정 매개변수가 SQL 변수가 아니라면, DB2는 지정 매개변수가 스키마 이름이라고 가정하고 이 스키마 이름을 특수 레지스터에 지정합니다.

SQL 프로시듀어에서 특수 레지스터 값의 초기 설정은 프로시듀어의 호출자로부터 계승됩니다. 새 설정값 지정은 그 값이 설정된 SQL 프로시듀어 전체에 유효하며 이 프로시듀어가 계속해서 호출하는 모든 프로시듀어에 의해 계승됩니다. 프로시듀어가 그의 호출자로 리턴하는 경우, 특수 레지스터는 호출자의 원래 설정값으로 복원됩니다.

expression 또는 **NULL**

지정 소스인 표현식이나 값을 지정합니다.

규칙

- SQL 프로시듀어의 지정 명령문은 SQL 지정 규칙에 따라야 합니다.
- 목표 및 소스의 데이터 유형은 호환 가능해야 합니다.
- 문자열이 고정 길이 변수에 지정되고 문자열의 길이가 목표의 길이 속성보다 작은 경우, 이 문자열의 오른쪽은 필요한 1바이트, 2바이트 또는 UCS-2 공백으로 채워집니다.
- 문자열이 변수에 지정되고 문자열이 변수의 길이 속성보다 긴 경우에는 오류가 발생합니다.
- 변수에 지정된 문자열은 필요할 경우 먼저 목표의 코드페이지로 변환됩니다.
- 숫자 변수에 지정시 해당 숫자 부분 전체가 절단되면 오류가 발생합니다.

예

SQL 변수 p_salary를 10 퍼센트만큼 증가시키십시오.

```
SET p_salary = p_salary + (p_salary * .10)
```

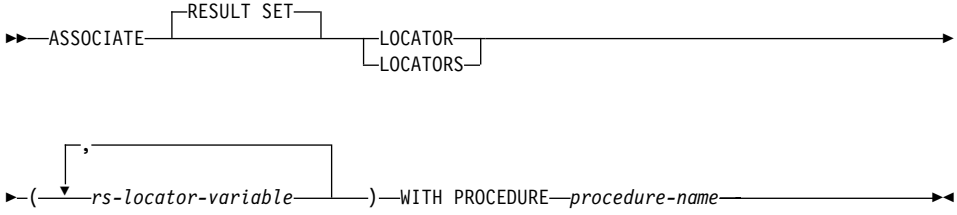
SQL 변수 p_salary를 널(NULL) 값으로 설정하십시오.

```
SET p_salary = NULL
```

ASSOCIATE LOCATORS문

ASSOCIATE LOCATORS 문은 저장 프로시듀어가 리턴하는 결과 세트 각각에 대한 결과 세트 위치 지정자 값을 가져옵니다.

구문



설명

rs-locator-variable

복합 텍스트 명령문에 선언된 결과 세트 위치 지정자 변수를 지정합니다.

WITH PROCEDURE

지정된 프로시듀어 이름으로 결과 세트 위치 지정자를 리턴하는 저장 프로시듀어를 식별합니다.

procedure-name

프로시듀어 이름은 규정된 이름 또는 규정되지 않은 이름입니다. 이름의 각 부분은 SBCS 문자들로 구성되어야 합니다.

완전한 프로시듀어 이름은 두 부분으로 구성된 이름입니다. 첫번째 부분은 저장 프로시듀어의 스키마 이름을 포함하는 식별자입니다. 마지막 부분은 저장 프로시듀어의 이름을 포함하는 식별자입니다. 각 부분은 마침표로 구분해야 합니다. 일부 또는 모든 부분은 분리 식별자일 수 있습니다.

프로시듀어 이름이 규정되지 않은 경우, 내재된 스키마 이름이 규정자로서 프로시듀어 이름에 추가되지 않기 때문에 유일하게 한 이름만을 가집니다. ASSOCIATE LOCATOR 문의 정상적인 실행을 위해서는 명령문의 프로시듀어 이름이 규정되지 않은 이름으로 지정된 가장 최근에 실행된 CALL 문의 프로시듀어 이름과 같아야 합니다. CALL 문의 규정되지 않

은 이름에 대한 내재된 스키마 이름은 일치하지 않아도 됩니다. 프로시듀어 이름 지정 방법에 대한 규칙이 아래에 설명되어 있습니다.

ASSOCIATE LOCATORS 문 실행시, 프로시듀어 이름이나 스펙은 리퀘스터가 이미 CALL 문을 사용하여 호출한 저장 프로시듀어를 식별해야 합니다. ASSOCIATE LOCATORS 문의 프로시듀어 이름은 CALL 문에 지정된 방법과 같은 방법으로 지정해야 합니다. 예를 들어 두 부분 이름을 CALL 문에 지정한 경우에는 ASSOCIATE LOCATORS 문에 두 부분 이름을 사용해야 합니다.

규칙

- 둘 이상의 위치 지정자를 결과 세트에 지정할 수 있습니다. 다른 결과 세트 위치 지정자 변수로 두 번 이상 같은 ASSOCIATE LOCATORS 문을 발행할 수 있습니다.
- ASSOCIATE LOCATORS 문에 나열된 결과 세트 위치 지정자 변수의 갯수가 저장 프로시듀어에서 리턴하는 위치 지정자 수보다 작은 경우, 이 명령문의 모든 변수에 하나의 값이 지정되고 경고가 발행됩니다.
- ASSOCIATE LOCATORS 문에 나열된 결과 세트 위치 지정자 변수의 갯수가 저장 프로시듀어에서 리턴하는 위치 지정자 수보다 큰 경우, 여분의 변수에 0 값이 지정됩니다.
- 같은 호출자가 저장 프로시듀어를 두 번 이상 호출하는 경우에는 가장 최근의 결과 세트만 액세스 가능합니다.

예

다음 예제의 명령문은 SQL 프로시듀어에 삽입된다고 가정합니다.

예 1: 결과 세트 위치 지정자 변수 LOC1과 LOC2를 사용하여 저장 프로시듀어 P1으로부터 리턴된 두 개의 결과 세트에 대한 결과 세트 위치 지정자 값을 가져오십시오. 저장 프로시듀어가 한 부분 이름으로 호출된다고 가정합니다.

```
CALL P1;
ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
   WITH PROCEDURE P1;
```

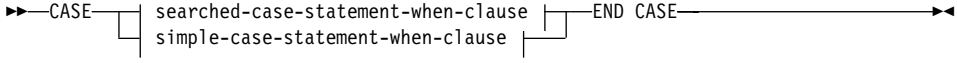
예 2: 예 1의 시나리오를 반복하되, 두 부분 이름을 사용하여 스키마 MYSCHEMA의 저장 프로시듀어 P1이 사용되도록 스키마에 대한 명시적 스키마 이름을 지정하십시오.

```
CALL MYSCHEMA.P1;
ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
   WITH PROCEDURE MYSCHEMA.P1;
```

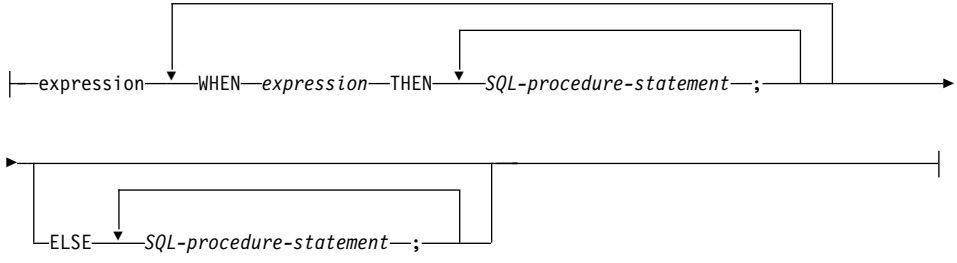
CASE문

CASE 문은 여러 조건을 근거로 실행 경로를 선택합니다.

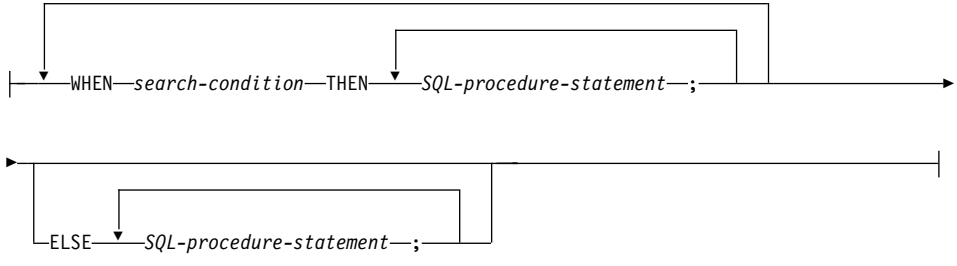
구문



simple-case-statement-when-clause:



searched-case-statement-when-clause:



설명

CASE

*case-expression*을 시작합니다.

simple-case-statement-when-clause

첫번째 WHEN 키워드 앞의 *expression* 값이 WHEN 키워드 다음에 오는 각 *expression* 값과 동일여부를 테스트됩니다. 검색 조건이 참인 경우 THEN 문

이 실행됩니다. 결과가 알 수 없거나 거짓인 경우 다음 검색 조건으로 처리가 계속됩니다. 결과가 검색 조건과 일치하지 않고 ELSE 절이 있는 경우 ELSE 절의 명령문이 처리됩니다.

searched-case-statement-when-clause

WHEN 키워드 다음의 *search-condition*이 평가됩니다. 참으로 평가되면 관련 THEN 절의 명령문이 처리됩니다. 거짓 또는 알 수 없음으로 평가되면 다음의 *search-condition*의 평가됩니다. 참으로 평가되는 *search-condition*이 없고 ELSE 절이 있는 경우에는 ELSE 절의 명령문이 처리됩니다.

SQL-procedure-statement

권한 취소해야 할 명령문을 지정합니다.

END CASE

*case-statement*를 끝마칩니다.

주

- WHEN에 지정된 조건 중 참인 조건이 없고 ELSE 절이 지정되지 않았다면 실행시 오류가 발생하고 CASE 문의 실행이 종료됩니다(SQLSTATE 20000).
- CASE 문이 가능한 모든 실행 조건을 변환하는지 확인하십시오.

예

SQL 변수 v_workdept의 값에 따라, DEPARTMENT 테이블의 컬럼 DEPTNAME을 적절한 이름으로 갱신하십시오.

다음 예는 *simple-case-statement-when-clause*에 대한 구문을 사용하여 수행하는 방법을 보여줍니다.

```
CASE v_workdept
   WHEN'A00'
     THEN UPDATE department
     SET deptname = 'DATA ACCESS 1';
   WHEN 'B01'
     THEN UPDATE department
     SET deptname = 'DATA ACCESS 2';
   ELSE UPDATE department
     SET deptname = 'DATA ACCESS 3';
END CASE
```

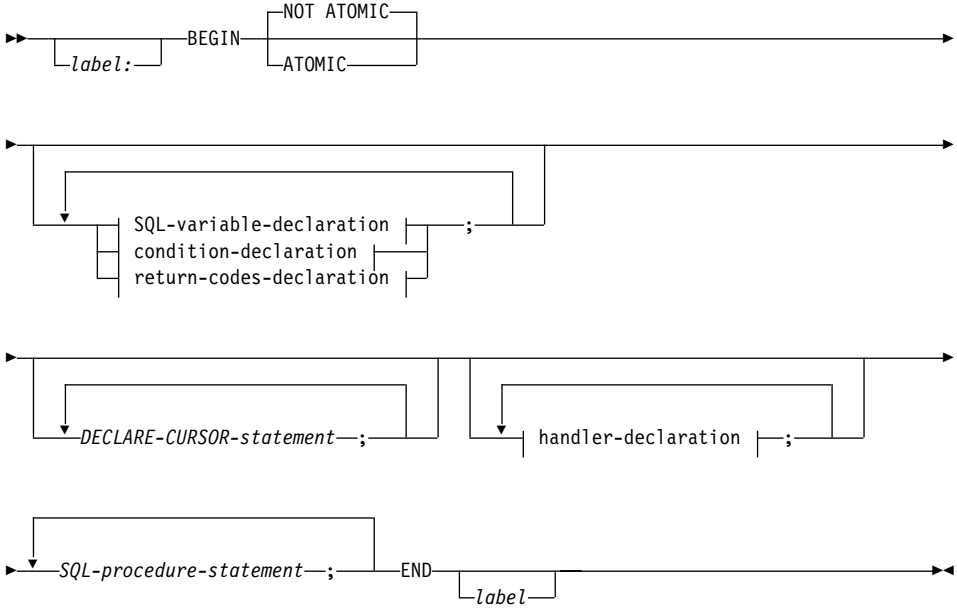
다음 예는 *searched-case-statement-when-clause*에 대한 구문을 사용하여 이를 수행하는 방법을 보여줍니다.

```
CASE
 WHEN v_workdept = 'A00'
   THEN UPDATE department
   SET deptname = 'DATA ACCESS 1';
 WHEN v_workdept = 'B01'
   THEN UPDATE department
   SET deptname = 'DATA ACCESS 2';
 ELSE UPDATE department
   SET deptname = 'DATA ACCESS 3';
END CASE
```

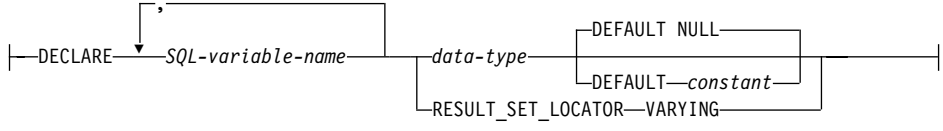
조합문

복합 텍스트 명령문은 SQL 프로시듀어에서 다른 명령문들을 함께 그룹으로 묶습니다. 복합 텍스트 명령문내에서 SQL 변수, 커서 및 조건 핸들러를 선언할 수 있습니다.

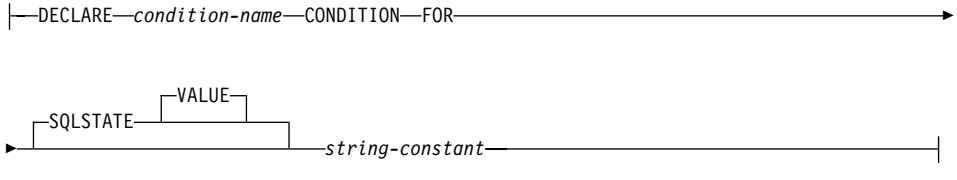
구문



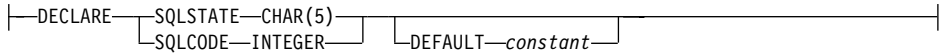
SQL-variable-declaration:



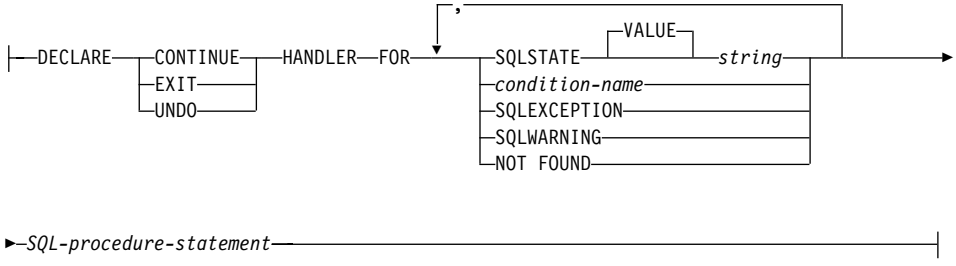
condition-declaration:



return-codes-declaration:



handler-declaration:



설명

label

코드 블록에 대한 레이블을 정의합니다. 시작 레이블이 지정되면, 이 레이블은 복합 텍스트 명령문에 선언된 SQL 변수를 규정하는 데 사용할 수 있고 LEAVE 문에 지정할 수도 있습니다. 끝 레이블이 지정되는 경우 이 레이블은 시작 레이블과 같아야 합니다.

ATOMIC 또는 NOT ATOMIC

ATOMIC은 복합 텍스트 명령문에서 오류가 발생하는 경우 복합 텍스트 명령문의 모든 SQL문이 구간 복원됨을 나타냅니다. NOT ATOMIC은 복합 텍스트 명령문내의 오류로 인해 복합 텍스트 명령문이 구간 복원되지 않음을 나타냅니다.

SQL-variable-declaration

복합 텍스트 명령문에 대한 지역 변수를 선언합니다.

SQL-variable-name

지역 변수의 이름을 정의합니다. DB2는 모든 SQL 변수 이름을 대문자로 변환합니다. 이 이름은 같은 복합 텍스트 명령문내의 다른 SQL 변수와 같을 수 없으며 매개변수 이름과 같을 수도 없습니다. SQL 변

수 이름은 컬럼 이름과 같아서는 안됩니다. SQL문에 SQL 변수 및 컬럼 참조와 같은 이름의 식별자가 포함되는 경우, DB2는 이 식별자를 컬럼으로 해석합니다.

data-type

변수의 데이터 유형을 지정합니다. SQL 데이터 유형의 설명에 대해서는 87 페이지의 『데이터 유형』을 참조하십시오. 사용자 정의 데이터 유형, 그래픽 유형 및 FOR BIT DATA는 지원되지 않습니다.

DEFAULT *constant* **또는 NULL**

SQL 변수의 기본값을 정의합니다. 변수는 SQL 프로시듀어가 호출될 때 초기화됩니다. 기본값이 지정되지 않을 경우 이 변수는 널(NULL)로 초기화됩니다.

RESULT_SET_LOCATOR VARYING

결과 세트 위치 지정자 변수의 데이터 유형을 지정합니다.

condition-declaration

조건 이름 및 해당되는 SQLSTATE 값을 선언합니다.

condition-name

조건 이름을 지정합니다. 조건 이름은 프로시듀어 본문에서 고유해야 하며 이 이름이 선언된 복합 텍스트 명령문내에서만 참조 가능합니다.

FOR SQLSTATE *string-constant*

조건과 연관된 SQLSTATE를 지정합니다. string-constant는 작은 따옴표로 묶인 5문자로 지정해야 하며 '00000'이 될 수 없습니다.

return-codes-declaration

SQL문 처리후 리턴된 값으로 자동 설정되는 특수 변수 SQLSTATE와 SQLCODE를 선언합니다. SQLSTATE와 SQLCODE 변수는 둘다 SQL 프로시듀어 본문의 가장 외부 복합 텍스트 명령문에서만 선언할 수 있습니다. 이러한 변수는 SQL 프로시듀어당 한번만 선언 가능합니다.

declare-cursor-statement

프로시듀어 본문에 커서를 선언합니다. 각 커서는 고유한 이름이어야 합니다. 커서는 복합 텍스트 명령문내에서만 참조할 수 있습니다. OPEN 문을 사용하여 커서를 열고 FETCH 문을 사용하여 커서로 행을 읽으십시오. 결과 세트를

SQL 프로시듀어에서 클라이언트 응용프로그램으로 리턴하려면, WITH RETURN 절을 사용하여 커서를 선언해야 합니다. 다음 예는 하나의 결과 세트를 클라이언트 응용프로그램으로 리턴합니다.

```
CREATE PROCEDURE RESULT_SET()
   LANGUAGE SQL
   RESULT SETS 1
   BEGIN
      DECLARE C1 CURSOR WITH RETURN FOR
         SELECT id, name, dept, job
            FROM staff;
      OPEN C1;
     END
```

주: 결과 세트를 처리하려면, DB2 콜 레벨 인터페이스(DB2 CLI), ODBC(Open Database Connectivity), JDBC(Java Database Connectivity) 또는 Java용 Embedded SQL(SQLJ) API(application programming interface) 중 하나를 사용하여 클라이언트 응용프로그램을 작성해야 합니다.

커서 선언에 대한 자세한 정보는 952 페이지의 『DECLARE CURSOR』에서 참조하십시오.

handler-declaration

복합 텍스트 명령문에서 예외 또는 완료 조건이 발생하는 경우 실행할 명령문 세트인 *핸들러*를 지정합니다. *SQL-procedure-statement*는 핸들러가 제어를 수신할 때 실행되는 명령문입니다.

핸들러는 이 핸들러가 선언된 복합 텍스트 명령문내에서만 활동 중입니다.

조건 핸들러의 유형에는 세 가지가 있습니다.

CONTINUE

핸들러가 정상적으로 호출된 후, 예외를 일으킨 명령문 다음에 오는 SQL 문으로 제어가 리턴됩니다. 예외를 일으킨 오류가 FOR, IF, CASE, WHILE 또는 REPEAT 문(이들 중 하나에 있는 SQL-procedure-statement는 아님)인 경우에는 제어가 END FOR, END IF, END CASE, END WHILE 또는 END REPEAT 다음에 오는 명령문으로 리턴됩니다.

EXIT

핸들러가 정상적으로 호출된 후, 핸들러를 선언한 복합 텍스트 명령문의 끝으로 제어가 리턴됩니다.

UNDO

핸들러가 호출되기 전에, 복합 텍스트 명령문에 작성된 모든 SQL 변경사항이 구간 복원됩니다. 핸들러가 정상적으로 호출된 후, 핸들러를 선언한 복합 텍스트 명령문의 끝으로 제어가 리턴됩니다. UNDO가 지정되면 ATOMIC을 지정해야 합니다.

핸들러가 활성화되는 조건은 다음과 같습니다.

SQLSTATE *string*

핸들러가 호출되는 SQLSTATE를 지정합니다. SQLSTATE는 '00000'이 될 수 없습니다.

condition-name

핸들러가 호출되는 조건 이름을 지정합니다. 이전에 해당 조건 이름을 조건 선언에 정의했어야 합니다.

SQLEXCEPTION

SQLEXCEPTION 발생시 핸들러가 호출되도록 지정합니다. SQLEXCEPTION은 처음 두 문자가 "00", "01" 또는 "02"가 아닌 SQLSTATE입니다.

SQLWARNING

SQLWARNING 발생시 핸들러가 호출되도록 지정합니다. SQLWARNING은 처음 두 문자가 "01"인 SQLSTATE입니다.

NOT FOUND

NOT FOUND 조건 발생시 핸들러가 호출되도록 지정합니다. NOT FOUND는 처음 두 문자가 "02"인 SQLSTATE에 해당됩니다.

규칙

- ATOMIC 조합문은 중첩될 수 없습니다.
- 다음 규칙이 핸들러 선언에 적용됩니다.

- SQLEXCEPTION, SQLWARNING 또는 NOT FOUND를 포함하는 핸들러 선언에는 추가 SQLSTATE나 조건 이름이 있을 수 없습니다.
- 같은 복합 텍스트 명령문내의 핸들러 선언에는 중복되는 조건이 포함될 수 없습니다.
- 핸들러 같음에는 같은 조건 코드나 SQLSTATE 값이 두 번 이상 포함될 수 없으며, 같은 SQLSTATE 값을 나타내는 조건 이름과 SQLSTATE 값이 포함될 수 없습니다. SQLSTATE 값 목록에 대해서는 *메시지 참조서*를 참조하십시오.
- 핸들러는 예외나 완료 조건에 가장 적합한 핸들러인 경우 활성화됩니다. 가장 적합한 핸들러는 범위에서 해당 예외 또는 완료 조건을 갖는 명령문에 가장 가깝게 복합 텍스트 명령문에서 정의된 핸들러(예외 또는 완료 조건에 대한)입니다. 핸들러 없음 예외가 발생하는 경우에는 복합 텍스트 명령문의 실행이 종료됩니다.

예

다음 조치를 수행하는 복합 텍스트 명령문으로 프로시듀어 본문을 작성하십시오.

1. SQL 변수 선언
2. 커서를 선언하여 IN 매개변수로 결정된 부서의 직원 급여를 리턴합니다. SELECT 문에서 *salary* 컬럼의 데이터 유형을 DECIMAL에서 DOUBLE로 변환합니다.
3. '6666' 값을 OUT 매개변수 medianSalary로 지정하는 조건 NOT FOUND(파일의 끝)에 대한 EXIT 핸들러를 선언합니다.
4. SQL 변수 numRecords에 지정된 부서의 직원 수를 선택하십시오.
5. 직원의 50% + 1이 검색될 때까지 WHILE 루프에서 커서의 행을 페치하십시오.
6. 평균 급여를 리턴하십시오.

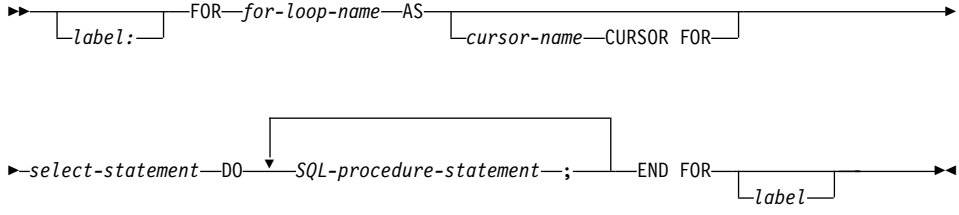
```
CREATE PROCEDURE DEPT_MEDIAN
   (IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
   LANGUAGE SQL
   BEGIN
    DECLARE v_numRecords INTEGER DEFAULT 1;
    DECLARE v_counter INTEGER DEFAULT 0;
    DECLARE c1 CURSOR FOR
```

```
          SELECT CAST(salary AS DOUBLE) FROM staff
             WHERE DEPT = deptNumber
             ORDER BY salary;
       DECLARE EXIT HANDLER FOR NOT FOUND
          SET medianSalary = 6666;
-- initialize OUT parameter
          SET medianSalary = 0;
          SELECT COUNT(*) INTO v_numRecords FROM staff
             WHERE DEPT = deptNumber;
          OPEN c1;
          WHILE v_counter < (v_numRecords / 2 + 1) DO
             FETCH c1 INTO medianSalary;
             SET v_counter = v_counter + 1;
          END WHILE;
          CLOSE c1;
        END
```

FOR문

FOR 문은 테이블의 각 행에 대해 하나의 명령문이나 명령문 그룹을 실행합니다.

구문



설명

label

FOR문에 대한 레이블을 지정합니다. 시작 레이블이 지정된 경우, 해당 레이블을 LEAVE 및 ITERATE 문에 사용할 수 있습니다. 끝 레이블이 지정된 경우 이 레이블은 시작 레이블과 같아야 합니다.

for-loop-name

FOR 문을 구현하기 위해 생성된 내재 복합 텍스트 명령문의 레이블을 지정합니다. FOR 문에서 ITERATE 또는 LEAVE 문과 함께 사용할 수 없다는 점을 제외하고는 복합 텍스트 명령문의 레이블 규칙을 따릅니다. *for-loop-name*은 지정된 *select-statement*에서 리턴하는 컬럼 이름을 규정하는 데 사용됩니다.

cursor-name

SELECT 문의 결과 테이블에서 행을 선택하는 데 사용되는 커서를 명명합니다. 지정되어 있지 않을 경우, DB2는 고유 커서 이름을 생성합니다.

select-statement

커서의 SELECT문을 지정합니다. 선택 목록의 모든 컬럼은 이름을 가져야 하며 같은 이름을 갖는 두 컬럼이 있을 수 없습니다.

SQL-procedure-statement

테이블의 각 행에 대해 호출되는 명령문(들)을 지정합니다.

규칙

- 선택 목록은 고유한 이름으로 구성되어야 하고 프로시듀어 작성시 선택 목록에 지정된 테이블이 존재해야 하거나 이전 SQL 프로시듀어 명령문으로 작성된 테이블이어야 합니다.
- FOR 문에 지정된 커서는 FOR 문 외부에서 참조될 수 없으며 OPEN, FETCH 또는 CLOSE 문에 지정될 수 없습니다.

예

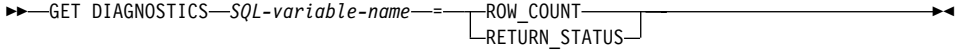
다음 예에서 FOR 문은 employee 테이블 전체를 반복하는 데 사용됩니다. 테이블의 각 행에 대해 SQL 변수 fullname이 직원의 성, 쉼표, 이름, 공백 하나 및 중간 이니셜로 설정됩니다. fullname에 대한 각 값은 tnames 테이블에 삽입됩니다.

```
BEGIN
   DECLARE fullname CHAR(40);
   FOR v1 AS
      SELECT firstnme, midinit, lastname FROM employee
         DO
         SET fullname = lastname || ',' || firstnme ||' ' || midinit;
         INSERT INTO tnames VALUE (fullname);
   END FOR
     END
```

GET DIAGNOSTICS문

GET DIAGNOSTICS 문은 호출된 이전 SQL문에 관한 정보를 가져옵니다.

구문



설명

SQL-variable-name

지정 목표인 변수를 식별합니다. 이 변수는 정수 변수이어야 합니다. SQL 변수는 복합 텍스트 명령문에 정의할 수 있습니다.

ROW_COUNT

호출된 이전 SQL문과 연관된 행의 수를 식별합니다. 이전 SQL문이 DELETE, INSERT 또는 UPDATE 문이라면, ROW_COUNT가 트리거나 참조 무결성 제한조건의 영향을 받는 행을 제외한 해당 명령문이 삭제, 삽입 또는 갱신한 행의 수를 식별합니다. 이전 명령문이 PREPARE 문인 경우, ROW_COUNT는 준비된 명령문에서 결과 행의 **추정** 갯수를 식별합니다.

RETURN_STATUS

해당 명령문이 상태를 리턴하는 프로시듀어를 호출하는 CALL 문이었을 경우 이전에 실행된 SQL문과 연관된 저장 프로시듀어로부터 리턴된 상태 값을 식별합니다. 이전 명령문이 그러한 명령문이 아닌 경에는 리턴된 값이 아무 의미도 없으며 임의의 정수일 수 있습니다.

규칙

- GET DIAGNOSTICS 문이 진단 영역(SQLCA)의 내용을 변경하지 않습니다. SQLSTATE 또는 SQLCODE 특수 변수가 SQL 프로시듀어에서 선언된 경우, 이러한 변수들은 발행되는 GET DIAGNOSTICS 문으로부터 리턴된 SQLSTATE 또는 SQLCODE로 설정됩니다.

예

SQL 프로시듀어에서 GET DIAGNOSTICS 문을 실행하여 갱신된 행의 수를 판별하십시오.

```
    CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3))
       LANGUAGE SQL
       BEGIN
          DECLARE SQLSTATE CHAR(5);
          DECLARE rcount INTEGER;
          UPDATE CORPDATA.PROJECT
  SET PRSTAFF = PRSTAFF + 1.5
             WHERE DEPTNO = deptnbr;
          GET DIAGNOSTICS rcount = ROW_COUNT;
-- At this point, rcount contains the number of rows that were updated.
    ...
         END
```

SQL 프로시듀어내에서 사용자 실패를 나타내는 양의 값을 명시적으로 리턴하거나 음의 상태 값을 리턴하는 SQL 오류를 발견할 수 있는 저장 프로시듀어 TRYIT의 호출로부터 리턴된 상태 값을 처리하십시오. 프로시듀어가 정상 수행되는 경우에는 0 값을 리턴합니다.

```
  CREATE PROCEDURE TESTIT ()
    LANGUAGE SQL
    A1:BEGIN
       DECLARE RETVAL INTEGER DEFAULT 0;
       ...
       CALL TRYIT;
       GET DIAGNOSTICS RETVAL = RETURN_STATUS;
       IF RETVAL <> 0 THEN
          ...
          LEAVE A1;
       ELSE
          ...
       END IF;
     END A1
```

GOTO문

GOTO 문은 SQL 루틴내에서 사용자 정의 레이블로의 브랜치에 사용됩니다.

구문

설명

label

처리가 계속되는 레이블이 붙은 명령문을 지정합니다. 레이블이 붙은 명령문과 GOTO 문은 같은 범위내에 있어야 합니다.

- GOTO 문이 FOR 문에 정의되는 경우, *label*은 중첩된 복합 텍스트 명령문을 제외한 동일한 FOR 문내에서 정의되어야 합니다.
- GOTO 문이 복합 텍스트 명령문에 정의된 경우, *label*은 중첩된 FOR문이나 중첩된 복합 텍스트 명령문을 제외한 동일한 복합 텍스트 명령문내에서 정의되어야 합니다.
- GOTO 문이 핸들러에 정의된 경우, *label*은 다른 범위 규칙에 따라 같은 핸들러에 정의되어야 합니다.
- GOTO 문이 핸들러 외부에서 정의된 경우, *label*은 핸들러내에서 정의되면 안됩니다.

*label*이 GOTO 문이 도달할 수 있는 범위내에서 정의되지 않은 경우에는 오류가 리턴됩니다(SQLSTATE 42736).

규칙

- GOTO 문을 너무 자주 사용하지 않도록 하십시오. 이 명령문은 정상적인 처리 순서를 방해하므로 루틴을 읽고 유지보수하는 데 보다 어렵게 합니다. GOTO 문을 사용하기 전에, IF 또는 LEAVE와 같은 다른 명령문을 사용하여 GOTO 문이 필요없게 할 수 있는지 여부를 판별하십시오.

예

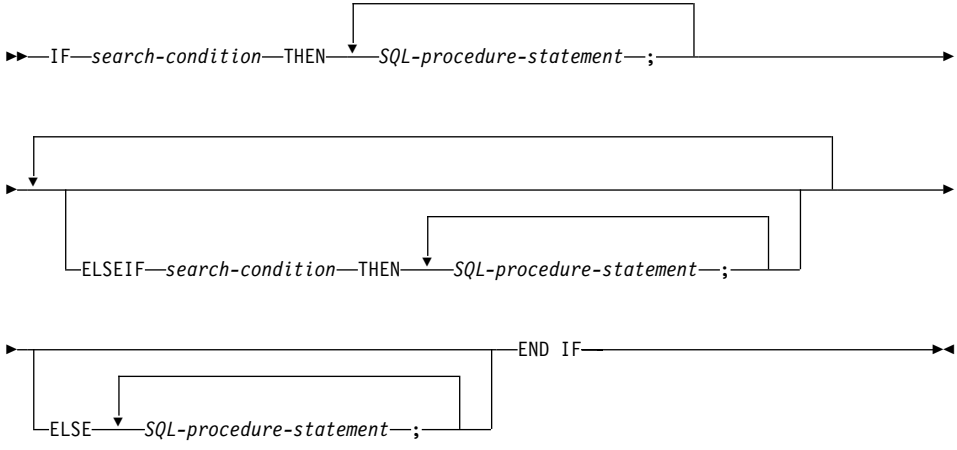
다음 복합 텍스트 명령문에서 *rating* 및 *v_empno* 매개변수가 프로시듀어로 전달되고, 이 프로시듀어는 출력 매개변수 *return_parm*을 날짜 기간으로 리턴합니다. 회사에서 직원의 근무 기간이 6개월 미만인 경우, GOTO 문이 제어를 프로시듀어의 끝으로 전달하고 *new_salary*는 변하지 않습니다.

```
     CREATE PROCEDURE adjust_salary
        (IN v_empno CHAR(6),
         IN rating INTEGER)
         OUT return_parm DECIMAL (8,2))
        MODIFIES SQL DATA
        LANGUAGE SQL
        BEGIN
           DECLARE new_salary DECIMAL (9,2)
           DECLARE service DECIMAL (8,2)
             SELECT SALARY, CURRENT_DATE - HIREDATE
               INTO new_salary, service
                             FROM EMPLOYEE
               WHERE EMPNO = v_empno
             IF service < 600
               THEN GOTO EXIT
             END IF
             IF rating = 1
               THEN SET new_salary = new_salary + (new_salary * .10)
             ELSE IF rating = 2
               THEN SET new_salary = new_salary + (new_salary * .05)
             END IF
UPDATE EMPLOYEE
               SET SALARY = new_salary
               WHERE EMPNO = v_empno
             EXIT: SET return_parm = service
          END
```

IF문

IF 문은 조건의 평가를 근거로 실행 경로를 선택합니다.

구문



설명

search-condition

SQL문을 호출해야 하는 조건을 지정합니다. 조건을 알 수 없거나 거짓인 경우, 조건이 참이거나 처리가 ELSE 절에 도달할 때까지 계속해서 다음 검색 조건으로 처리됩니다.

SQL-procedure-statement

이전 검색 조건이 참인 경우 호출될 명령문을 지정합니다. 참으로 평가되는 search-condition이 없는 경우에는 ELSE 키워드 다음의 *SQL-procedure-statement*이 호출됩니다.

예

다음 SQL 프로시듀어는 두 개의 IN 매개변수를 수용합니다. 직원 번호 *employee_number* 및 직위 *rating*. *rating* 값에 따라, `employee` 테이블은 `salary` 및 `bonus` 컬럼의 새 값으로 갱신됩니다.

```
CREATE PROCEDURE UPDATE_SALARY_IF
   (IN employee_number CHAR(6), INOUT rating SMALLINT)
   LANGUAGE SQL
   BEGIN
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE EXIT HANDLER FOR not_found
         SET rating = -1;
      IF rating = 1
         THEN UPDATE employee
         SET salary = salary * 1.10, bonus = 1000
         WHERE empno = employee_number;
      ELSEIF rating = 2
         THEN UPDATE employee
         SET salary = salary * 1.05, bonus = 500
         WHERE empno = employee_number;
      ELSE UPDATE employee
         SET salary = salary * 1.03, bonus = 0
         WHERE empno = employee_number;
      END IF;
     END
```

ITERATE문

ITERATE 문은 제어 흐름을 레이블이 붙은 루프의 시작부분으로 리턴합니다.

구문

설명

label

DB2가 제어 흐름을 전달하는 FOR, LOOP, REPEAT 또는 WHILE 문을 지정합니다.

예

이 예는 커서를 사용하여 새 부서에 대한 정보를 리턴합니다. *not_found* 조건 핸들러가 호출되는 경우 제어 흐름이 루프 밖으로 전달됩니다. *v_dept* 값이 'D11'이라면 ITERATE 문이 제어 흐름을 다시 LOOP 문의 맨 위로 전달합니다. 그렇지 않으면 새 행이 DEPARTMENT 테이블에 삽입됩니다.

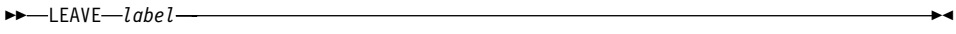
```
CREATE PROCEDURE ITERATOR()
   LANGUAGE SQL
   BEGIN
      DECLARE v_dept CHAR(3);
      DECLARE v_deptname VARCHAR(29);
      DECLARE v_admdept CHAR(3);
      DECLARE at_end INTEGER DEFAULT 0;
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE c1 CURSOR FOR
         SELECT deptno, deptname, admrdept
         FROM department
         ORDER BY deptno;
      DECLARE CONTINUE HANDLER FOR not_found
         SET at_end = 1;
      OPEN c1;
      ins_loop:
      LOOP
         FETCH c1 INTO v_dept, v_deptname, v_admdept;
         IF at_end = 1 THEN
            LEAVE ins_loop;
         ELSEIF v_dept = 'D11' THEN
            ITERATE ins_loop;
```

```
    END IF;
    INSERT INTO department (deptno, deptname, admrdept)
    VALUES ('NEW', v_deptname, v_admdept);
 END LOOP;
 CLOSE c1;
END
```

LEAVE문

LEAVE 문은 프로그램 제어를 루프나 복합 텍스트 명령문 밖으로 전송합니다.

구문



설명

label

나가기 위해 복합 텍스트, FOR, LOOP, REPEAT 또는 WHILE 문의 레이블을 지정합니다.

규칙

LEAVE 문이 제어를 복합 텍스트 밖으로 전송하는 경우, 결과 세트를 리턴하는 데 사용되는 커서를 제외한 복합 텍스트 명령문내의 열려 있는 모든 커서가 닫힙니다.

예

이 예에는 *c1* 커서에 대한 데이터를 페치하는 루트가 포함됩니다. SQL 변수 *at_end*의 값이 0이 아닌 경우, LEAVE 문은 제어를 루프 밖으로 전송합니다.

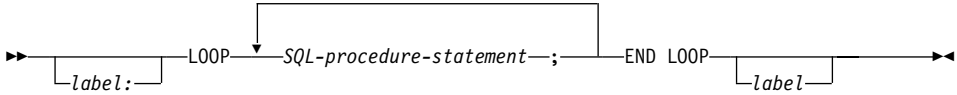
```
CREATE PROCEDURE LEAVE_LOOP(OUT counter INTEGER)
   LANGUAGE SQL
   BEGIN
      DECLARE v_counter INTEGER;
      DECLARE v_firstnme VARCHAR(12);
      DECLARE v_midinit CHAR(1);
      DECLARE v_lastname VARCHAR(15);
      DECLARE at_end SMALLINT DEFAULT 0;
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE c1 CURSOR FOR
         SELECT firstnme, midinit, lastname
            FROM employee;
      DECLARE CONTINUE HANDLER for not_found
         SET at_end = 1;
      SET v_counter = 0;
      OPEN c1;
      fetch_loop:
```

```
 LOOP
    FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
    IF at_end <> 0 THEN LEAVE fetch_loop;
    END IF;
    SET v_counter = v_counter + 1;
 END LOOP fetch_loop;
 SET counter = v_counter;
 CLOSE c1;
END
```

LOOP문

LOOP 문은 명령문이나 명령문 그룹의 실행을 반복합니다.

구문



설명

label

LOOP 문에 대한 레이블을 지정합니다. 시작 레이블이 지정된 경우, 해당 레이블을 LEAVE 및 ITERATE 문에 지정할 수 있습니다. 끝 레이블이 지정된 경우 일치하는 시작 레이블을 지정해야 합니다.

SQL-procedure-statement

루프에서 호출될 명령문을 지정합니다.

예

이 프로시듀어는 LOOP 문을 사용하여 employee 테이블의 값을 페치합니다. 루프가 반복될 때마다, OUT 매개변수 *counter*가 증가되고 *v_midinit* 값이 하나의 공백(' ')여부를 확인합니다. *v_midinit*가 하나의 공백이라면 LEAVE 문이 제어 흐름을 루프 밖으로 전달합니다.

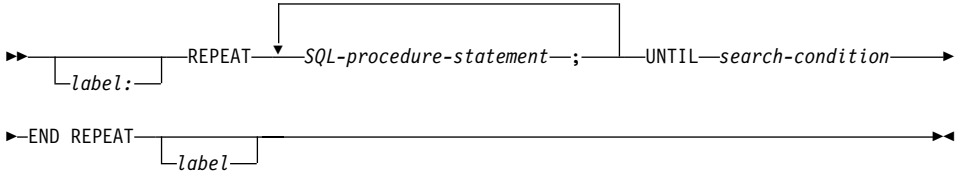
```
CREATE PROCEDURE LOOP_UNTIL_SPACE(OUT counter INTEGER)
   LANGUAGE SQL
   BEGIN
      DECLARE v_counter INTEGER DEFAULT 0;
      DECLARE v_firstnme VARCHAR(12);
      DECLARE v_midinit CHAR(1);
      DECLARE v_lastname VARCHAR(15);
      DECLARE c1 CURSOR FOR
         SELECT firstnme, midinit, lastname
            FROM employee;
      DECLARE CONTINUE HANDLER FOR NOT FOUND
         SET counter = -1;
      OPEN c1;
      fetch_loop:
      LOOP
```

```
    FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
    IF v_midinit = ' ' THEN
       LEAVE fetch_loop;
    END IF;
    SET v_counter = v_counter + 1;
 END LOOP fetch_loop;
 SET counter = v_counter;
 CLOSE c1;
END
```

REPEAT문

REPEAT 문은 검색 조건이 참이 될 때까지 하나의 명령문이나 명령문 그룹을 실행합니다.

구문



설명

label

REPEAT 문에 대한 레이블을 지정합니다. 시작 레이블이 지정된 경우, 해당 레이블을 LEAVE 및 ITERATE 문에 지정할 수 있습니다. 끝 레이블이 지정된 경우 일치하는 시작 레이블도 지정해야 합니다.

SQL-procedure-statement

루프에서 실행될 SQL문을 지정합니다.

search-condition

SQL 프로시듀어 명령문의 각 실행 전에 평가되는 조건을 지정합니다. 조건이 거짓이라면 DB2가 해당 루프에서 SQL 프로시듀어 명령문을 실행합니다.

예

REPEAT 문은 *not_found* 조건 핸들러가 호출될 때까지 테이블에서 행을 페치합니다.

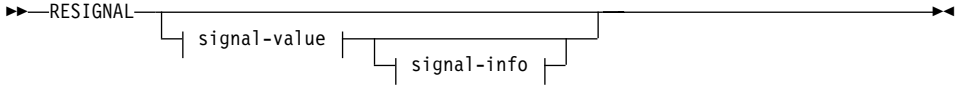
```
CREATE PROCEDURE REPEAT_STMT(OUT counter INTEGER)
   LANGUAGE SQL
   BEGIN
     DECLARE v_counter INTEGER DEFAULT 0;
     DECLARE v_firstnme VARCHAR(12);
     DECLARE v_midinit CHAR(1);
     DECLARE v_lastname VARCHAR(15);
     DECLARE at_end SMALLINT DEFAULT 0;
```

```
DECLARE not_found CONDITION FOR SQLSTATE '02000';
DECLARE c1 CURSOR FOR
   SELECT firstnme, midinit, lastname
      FROM employee;
DECLARE CONTINUE HANDLER FOR not_found
   SET at_end = 1;
OPEN c1;
fetch_loop:
REPEAT
   FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
   SET v_counter = v_counter + 1;
   UNTIL at_end > 0
END REPEAT fetch_loop;
SET counter = v_counter;
CLOSE c1;
END
```

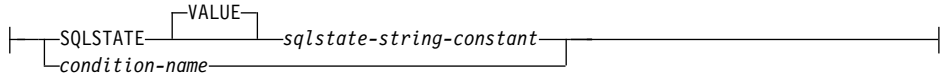
RESIGNAL문

RESIGNAL 문은 오류 또는 경고 조건의 신호를 다시 보내는 데 사용됩니다. 이로 인해 선택적 메시지 텍스트, 지정된 SQLSTATE와 함께 오류나 경고가 리턴됩니다.

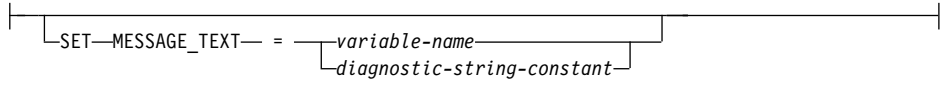
구문



signal-value:



signal-info:



설명

SQLSTATE VALUE *sqlstate-string-constant*

지정된 문자열 상수는 SQLSTATE를 나타냅니다. 이는 SQLSTATE 규칙을 따르는 정확히 5자로 된 문자열 상수이어야 합니다.

- 각 문자는 숫자('0' - '9') 집합이거나 강조되지 않은 대문자('A' through 'Z') 이어야 합니다.
- SQLSTATE 클래스(처음 두 문자)는 '00'(성공적인 완료를 나타냄)일 수 없습니다.

SQLSTATE가 이 규칙에 따르지 않으면 오류가 발생합니다(SQLSTATE 428B3).

condition-name

조건 이름을 지정합니다.

SET MESSAGE_TEXT =

오류나 경고를 설명하는 문자열을 지정합니다. 이 문자열은 SQLCA의 SQLERRMC 필드에 리턴됩니다. 실제 문자열이 70바이트보다 큰 경우에는 경고없이 절단됩니다. 이 절은 SQLSTATE 또는 *condition-name*도 지정된 경우에만 지정할 수 있습니다(SQLSTATE 42601).

variable-name

복합 텍스트 명령문내에서 선언해야 하는 SQL 변수를 식별합니다. SQL 변수는 CHAR 또는 VARCHAR 데이터 유형으로 정의해야 합니다.

diagnostic-string-constant

메시지 텍스트를 포함한 문자열 상수를 지정합니다.

주

- SQLSTATE 절이나 *condition-name*없이 RESIGNAL 문을 지정하는 경우, SQL 루틴은 핸들러를 호출한 동일한 조건으로 호출자에 리턴됩니다.
- RESIGNAL문이 발행되고 SQLSTATE 또는 *condition-name*이 지정된 경우, SQLCODE를 다음과 같이 지정합니다.

  SQLSTATE가 '01' 또는 '02'와 함께 시작할 경우에는 +438.

  그렇지 않은 경우에는 -438.

  RESIGNAL 문이 옵션없이 발행된 경우, SQLCODE는 핸들러가 호출되게 하는 예외로부터 변경되지 않습니다.
- SQLSTATE 또는 조건이 예외 신호 전달을 나타내는 경우('01' 또는 '02'가 아닌 SQLSTATE 클래스),
  - 핸들러가 RESIGNAL 문으로 핸들러를 포함하는 복합 텍스트 명령문으로부터 그 다음의 외부 복합 텍스트 명령문(또는 더 외부에 있는 복합 텍스트 명령문)에 있고 복합 텍스트 명령문에 지정된 SQLSTATE, condition-name 또는 SQLEXCEPTION에 대한 핸들러가 포함된다면 해당 예외가 처리되고 제어가 핸들러로 전송됩니다.
  - 그렇지 않으면 예외가 처리되지 않고 제어는 즉시 복합 텍스트 명령문의 끝으로 리턴됩니다.

- SQLSTATE 또는 조건이 경고(SQLSTATE 클래스 '01') 또는 찾을 수 없음 조건(SQLSTATE 클래스 '02')이 신호 전달됨을 나타냅니다.
  - 핸들러가 RESIGNAL 문으로 핸들러를 포함하는 복합 텍스트 명령문으로부터 그 다음의 외부 복합 텍스트 명령문(또는 더 외부에 있는 복합 텍스트 명령문)에 있고 복합 텍스트 명령문에 지정된 SQLSTATE, condition-name, SQLWARNING(SQLSTATE 클래스가 '01'인 경우) 또는 NOT FOUND(SQLSTATE 클래스가 '02'인 경우)에 대한 핸들러가 포함된다면 경고 또는 찾을 수 없음 조건이 처리되고 제어가 핸들러로 전송됩니다.
  - 그렇지 않으면 경고가 처리되지 않고 다음 명령문으로 처리는 계속됩니다.

예

이 예는 0으로 나눔 오류를 발견합니다. IF 문은 SIGNAL 문을 사용하여 *overflow* 조건 핸들러를 호출합니다. 조건 핸들러는 RESIGNAL 문을 사용하여 다른 SQLSTATE 값을 클라이언트 응용프로그램으로 리턴합니다.

```
CREATE PROCEDURE divide ( IN numerator INTEGER
                          IN denominator INTEGER
                          OUT result INTEGER
LANGUAGE SQL
CONTAINS SQL
BEGIN
  DECLARE overflow CONDITION FOR SQLSTATE '22003';
    DECLARE CONTINUE HANDLER FOR overflow
    RESIGNAL SQLSTATE'22375' ;
    IF denominator = 0 THEN
    SIGNAL overflow;
  ELSE
     SET result = numerator / denominator;
  END IF;
       END
```

RETURN문

RETURN 문은 루틴으로부터 리턴하는 데 사용됩니다. SQL 함수 또는 메소드의 경우에는 함수나 메소드의 결과를 리턴합니다. SQL 프로시듀어의 경우 선택적으로 정수 상태 값을 리턴합니다.

구문

설명

expression

루틴으로부터 리턴된 값을 지정합니다.

- 루틴이 함수이거나 메소드인 경우, *expression*을 지정해야 하고(SQLSTATE 42630) *expression*의 데이터 유형이 루틴의 RETURNS 유형에 지정되어야 합니다(SQLSTATE 42866).
- 루틴이 프로시듀어인 경우, *expression*의 데이터 유형은 INTEGER이어야 합니다(SQLSTATE 428E2).

주

- 값이 프로시듀어로부터 리턴되는 경우 호출자는 다음을 사용하여 값에 액세스할 수 있습니다.
  - GET DIAGNOSTICS 문을 사용하여 SQL 프로시듀어가 다른 SQL 프로시듀어로부터 호출될 때 RETURN_STATUS 검색
  - CLI 응용프로그램에서 escape 절 CALL 구문 (?=CALL...)의 리턴 값 매개변수 표시문자에 대해 바인드된 매개변수
  - SQLCODE가 0보다 작지 않은 경우(SQLCODE가 0보다 작은 경우에는 -1 값 가정) SQLERRD[0] 값을 검색함으로써 SQL 프로시듀어의 CALL 처리로부터 리턴된 SQLCA에서 직접.

예

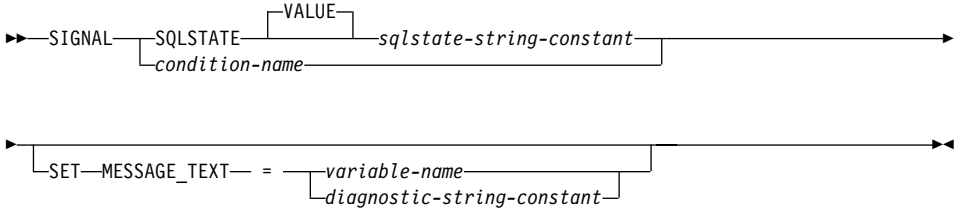
RETURN 문을 사용하여 정상적인 경우 상태 값 0과 그렇지 않은 경우 -200으로 SQL 저장 프로시듀어에서 리턴하십시오.

```
BEGIN
...
   GOTO FAIL
...
   SUCCESS: RETURN 0
   FAIL: RETURN -200
       END
```

SIGNAL문

SIGNAL 문은 오류 또는 경고 조건의 신호를 보내는 데 사용됩니다. 이로 인해 선택적 메시지 텍스트, 지정된 SQLSTATE와 함께 오류나 경고가 리턴됩니다.

구문



설명

SQLSTATE VALUE sqlstate-string-constant

지정된 문자열 상수는 SQLSTATE를 나타냅니다. 이는 SQLSTATE 규칙을 따르는 정확히 5자로 된 문자열 상수이어야 합니다.

- 각 문자는 숫자('0' - '9') 집합이거나 강조되지 않은 대문자('A' through 'Z') 이어야 합니다.
- SQLSTATE 클래스(처음 두 문자)는 '00'(성공적인 완료를 나타냄)일 수 없습니다.

SQLSTATE가 이 규칙에 따르지 않으면 오류가 발생합니다(SQLSTATE 428B3).

condition-name

조건 이름을 지정합니다. 조건 이름은 프로시듀어에서 고유해야 하며 이 이름이 선언된 복합 텍스트 명령문내에서만 참조 가능합니다.

SET MESSAGE_TEXT=

오류나 경고를 설명하는 문자열을 지정합니다. 이 문자열은 SQLCA의 SQLERRMC 필드에 리턴됩니다. 실제 문자열이 70바이트보다 큰 경우에는 경고없이 절단됩니다. 이 절은 SQLSTATE 또는 condition-name도 지정된 경우에만 지정할 수 있습니다(SQLSTATE 42601).

variable-name

복합 텍스트 명령문내에서 선언해야 하는 SQL 변수를 식별합니다. SQL 변수는 CHAR 또는 VARCHAR 데이터 유형으로 정의해야 합니다.

diagnostic-string-constant

메시지 텍스트를 포함한 문자열 상수를 지정합니다.

주

- SIGNAL문이 발행될 경우, SQLCODE는 다음과 같이 지정됩니다.

  SQLSTATE가 '01' 또는 '02'와 함께 시작할 경우에는 +438.

  그렇지 않은 경우에는 -438.

- SQLSTATE 또는 조건이 예외 신호 전달을 나타내는 경우('01' 또는 '02'가 아닌 SQLSTATE 클래스),
  - 핸들러가 SIGNAL 문과 같은 복합 텍스트 명령문(또는 외부 복합 텍스트 명령문)에 있고 복합 텍스트 명령문에 지정된 SQLSTATE, condition-name, SQLEXCEPTION에 대한 핸들러가 포함된다면 예외가 처리되고 제어가 핸들러로 전송됩니다.
  - 그렇지 않으면 예외가 처리되지 않고 제어는 즉시 복합 텍스트 명령문의 끝으로 리턴됩니다.
- SQLSTATE 또는 조건이 경고(SQLSTATE 클래스 '01') 또는 찾을 수 없음 조건(SQLSTATE 클래스 '02')의 신호 전달을 나타내는 경우,
  - 핸들러가 SIGNAL 문과 같은 복합 텍스트 명령문(또는 외부 복합 텍스트 명령문)에 있고 복합 텍스트 명령문에 지정된 SQLSTATE, condition-name, SQLWARNING(SQLSTATE 클래스가 '01'인 경우) 또는 NOT FOUND(SQLSTATE가 '02'인 경우)에 대한 핸들러가 포함된다면 경고 또는 찾을 수 없음 조건이 처리되고 제어가 핸들러로 전송됩니다.
  - 그렇지 않으면 경고가 처리되지 않고 다음 명령문으로 처리는 계속됩니다.
- SQLSTATE 값은 두 문자 클래스 코드 값과 세 문자 서브클래스 코드 값으로 구성됩니다. 클래스 코드 값은 성공 및 실패 실행 조건의 클래스를 나타냅니다.

  유효한 모든 SQLSTATE 값은 SIGNAL문에서 사용될 수 있습니다. 그러나 프로그래머는 응용프로그램용으로 예약된 범위를 근거로 새 SQLSTATE를 정의

하는 것이 바람직합니다. 이를 통해 차후 릴리스에서 데이터베이스 관리 프로그램이 잘못된 SQLSTATE 값을 정의하는 것을 막을 수 있습니다.

- '7' - '9' 또는 'I' - 'Z' 문자로 시작하는 SQLSTATE 클래스를 정의할 수 있습니다. 이들 클래스 내에 있는 모든 서브클래스가 정의됩니다.
- '0' - '6' 또는 'A' - 'H' 문자로 시작하는 SQLSTATE 클래스는 데이터베이스 관리 프로그램용으로 예약되어 있습니다. 이러한 클래스에서는 '0' - 'H' 문자로 시작하는 서브클래스가 데이터베이스 관리 프로그램용으로 예약되어 있습니다. 'I' - 'Z' 문자로 시작하는 서브클래스를 정의할 수 있습니다.

예

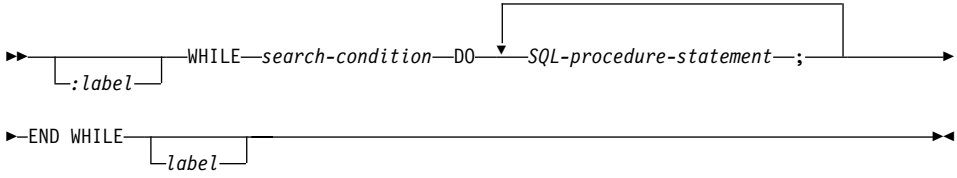
응용프로그램에 고객 번호가 알려져 있지 않은 경우 응용프로그램 오류 신호를 보내는 주문 시스템용 SQL 프로시듀어. ORDERS 테이블에는 CUSTOMER 테이블에 대한 외부 키가 포함되며, 주문을 삽입하기 위해서는 CUSTNO가 필요합니다.

```
CREATE PROCEDURE SUBMIT_ORDER
   (IN ONUM INTEGER, IN CNUM INTEGER,
    IN PNUM INTEGER, IN QNUM INTEGER)
    SPECIFIC SUBMIT_ORDER
    MODIFIES SQL DATA
   LANGUAGE SQL
   BEGIN
      DECLARE EXIT HANDLER FOR SQLSTATE VALUE '23503'
         SIGNAL SQLSTATE '75002'
            SET MESSAGE_TEXT = 'Customer number is not known';
      INSERT INTO ORDERS (ORDERNO, CUSTNO, PARTNO, QUANTITY)
         VALUES (ONUM, CNUM, PNUM, QNUM);
       END
```

WHILE문

WHILE 문은 지정된 조건이 참인 동안 하나의 명령문이나 명령문 그룹의 실행을 반복합니다.

구문



설명

label

WHILE 문에 대한 레이블을 지정합니다. 시작 레이블이 지정된 경우, 해당 레이블을 LEAVE 및 ITERATE 문에 지정할 수 있습니다. 끝 레이블이 지정된 경우 이 레이블은 시작 레이블과 같아야 합니다.

search-condition

루프의 각 실행 전에 평가되는 조건을 지정합니다. 조건이 참이라면 루프내의 SQL-procedure-statement가 처리됩니다.

SQL-procedure-statement

루프내에서 실행될 SQL문(들)을 지정합니다.

예

이 예는 WHILE 문을 사용하여 FETCH와 SET 문을 반복합니다. SQL 변수 *v_counter*의 값이 IN 매개변수 *deptNumber*로 식별되는 부서내 직원 수의 절반보다 작은 경우, WHILE 문은 계속해서 FETCH 및 SET 문을 반복합니다. 조건이 더 이상 참이 아니라면 제어 흐름이 WHILE 문을 나가서 커서를 닫습니다.

```
CREATE PROCEDURE DEPT_MEDIAN
   (IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
   LANGUAGE SQL
   BEGIN
      DECLARE v_numRecords INTEGER DEFAULT 1;
```

```
DECLARE v_counter INTEGER DEFAULT 0;
DECLARE c1 CURSOR FOR
   SELECT CAST(salary AS DOUBLE)
      FROM staff
      WHERE DEPT = deptNumber
      ORDER BY salary;
DECLARE EXIT HANDLER FOR NOT FOUND
   SET medianSalary = 6666;
SET medianSalary = 0;
SELECT COUNT(*) INTO v_numRecords
      FROM staff
      WHERE DEPT = deptNumber;
OPEN c1;
WHILE v_counter < (v_numRecords / 2 + 1) DO
   FETCH c1 INTO medianSalary;
   SET v_counter = v_counter + 1;
END WHILE;
CLOSE c1;
END
```

부록A. SQL 한계

다음 표에는 특정의 SQL 한계가 나와 있습니다. 프로그래머가 가장 제한적인 경우를 따른다면 쉽게 이식할 수 있는 응용프로그램을 설계할 수 있습니다.

표 29. 식별자 길이 한계

	설명	한계(바이트)
1	권한 부여 이름의 최대 길이(1바이트 문자만 가능)	30
2	제한조건 이름의 최대 길이	18
3	상관 이름의 최대 길이	128
4	조건 이름의 최대 길이	64
5	커서 이름의 최대 길이	18
6	데이터 소스 컬럼 이름의 최대 길이	128
7	데이터 소스 색인 이름의 최대 길이	128
8	데이터 소스 이름의 최대 길이	128
9	데이터 소스 테이블 이름의 최대 길이	128
10	외부 프로그램 이름의 최대 길이	8
11	호스트 식별자의 최대 길이[a]	255
12	데이터 소스 사용자 식별자의 최대 길이	30
13	레이블 이름의 최대 길이	64
14	메소드 이름의 최대 길이	18
15	매개변수 이름의 최대 길이[b]	128
16	데이터 소스를 액세스하기 위한 암호의 최대 길이	32
17	가장 긴 세이브포인트 이름	128
18	스키마 이름의 최대 길이[c]	30
19	서버(데이터베이스 별명) 이름의 최대 길이	8
20	SQL 변수 이름의 최대 길이	64
21	명령문 이름의 최대 길이	18
22	변환 그룹 이름의 최대 길이	18
23	규정화되지 않은 컬럼 이름의 최대 길이	30
24	규정화되지 않은 패키지 이름의 최대 길이	8

표 29. 식별자 길이 한계 (계속)

	설명	한계(바이트)
25	규정화되지 않은 사용자 정의 유형, 사용자 정의 함수, 버퍼 풀, 테이블 공간, 노드 그룹, 트리거, 색인 또는 색인 스펙 이름의 최대 길이	18
26	규정화되지 않은 테이블 이름, 뷰 이름, 저장 프로시듀어, 별명 또는 별명의 최대 길이	128
27	가장 긴 랩퍼 이름	128

주:

a 개별적 호스트 언어 컴파일러는 변수 이름에 대해 더 많은 제한을 둘 수 있습니다.

b SQL 프로시듀 내의 매개변수 이름은 64바이트가 한계입니다.

c 사용자 정의 구조화 유형에 대한 스키마 이름은 8바이트가 한계입니다.

표 30. 수치 한계

	설명	한계
1	최소 INTEGER 값	-2 147 483 648
2	최대 INTEGER 값	+2 147 483 647
3	최소 BIGINT 값	-9 223 372 036 854 775 808
4	최대 BIGINT 값	+9 223 372 036 854 775 807
5	최소 SMALLINT 값	-32 768
6	최대 SMALLINT 값	+32 767
7	최대 십진수(decimal) 정밀도	31
8	최소 DOUBLE 값	-1.79769E+308
9	최대 DOUBLE 값	+1.79769E+308
10	최소 양수 DOUBLE 값	+2.225E-307
11	최대 음수 DOUBLE 값	-2.225E-307
12	최소 REAL 값	-3.402E+38
13	최대 REAL 값	+3.402E+38
14	양의 최소 REAL 값	+1.175E-37
15	음의 최대 REAL 값	-1.175E-37

표 31. 문자열 한계

	설명	한계
1	CHAR 최대 길이(바이트 단위)	254
2	VARCHAR 최대 길이(바이트 단위)	32 672
3	LONG VARCHAR 최대 길이(바이트 단위)	32 700
4	CLOB 최대 길이(바이트 단위)	2 147 483 647
5	GRAPHIC 최대 길이(바이트 단위)	127
6	VARGRAPHIC 최대 길이(문자)	16 336
7	LONG VARGRAPHIC 최대 길이(문자)	16 350
8	DBCLOB 최대 길이(문자)	1 073 741 823
9	BLOB 최대 길이(바이트 단위)	2 147 483 647
10	문자 상수 최대 길이	32 672
11	그래픽 상수 최대 길이	16 336
12	병합 문자열 최대 길이	2 147 483 647
13	병합 그래픽 문자열 최대 길이	1 073 741 823
14	병합 2진 문자열 최대 길이	2 147 483 647
15	16진 상수 최대 자리수	16 336
16	카탈로그 주석 최대 크기(바이트 단위)	254
17	수행시 구조화 유형 컬럼 오브젝트의 가장 긴 인스턴스	1 GB

표 32. 날짜/시간 한계

	설명	한계
1	최소 DATE 값	0001-01-01
2	최대 DATE 값	9999-12-31
3	최소 TIME 값	00:00:00
4	최대 TIME 값	24:00:00
5	최소 TIMESTAMP 값	0001-01-01-00.00.00.000000
6	최대 TIMESTAMP 값	9999-12-31-24.00.00.000000

표 33. 데이터베이스 관리 프로그램 한계

	설명	한계
1	테이블의 대부분의 컬럼 [g]	1 012
2	뷰의 최대 컬럼 [a]	5 000
3	모든 오버헤드를 포함한 행의 최대 길이 [b] [g]	32 677

표 33. 데이터베이스 관리 프로그램 한계 (계속)

	설명	한계
4	파티션당 최대 테이블 크기(기가바이트) [c] [g]	512
5	파티션당 최대 색인 크기(기가바이트)	512
6	파티션당 테이블의 최대 행	4×10^9
7	모든 오버헤드를 포함하는 최장 색인 키(바이트)	1 024
8	색인 키의 최대 컬럼	16
9	테이블의 최대 색인	32 767 또는 저장영역
10	SQL문 또는 뷰(view)에서 참조되는 최대 테이블	저장영역
11	사전 처리 컴파일 프로그램의 최대 호스트 변수 선언 [c]	저장영역
12	SQL문의 최대 호스트 변수 참조	32 767
13	삽입 또는 갱신에 사용되는 최장 호스트 변수 값(바이트)	2 147 483 647
14	최장 SQL문(바이트)	65 535
15	선택 목록의 최대 요소[g]	1 012
16	WHERE 또는 HAVING절의 최대 술어	저장영역
17	GROUP BY절의 최대 컬럼 수 [g]	1 012
18	GROUP BY절의 최대 총 길이(바이트)[g]	32 677
19	ORDER BY절의 최대 컬럼 수 [g]	1 012
20	ORDER BY절의 최대 총 컬럼 길이(바이트) [g]	32 677
21	SQLDA의 최대 크기(바이트)	저장영역
22	준비된 명령문의 최대 수	저장영역
23	프로그램에 선언된 최대 커서 수	저장영역
24	한 번에 열리는 최대 커서 수	저장영역
25	SMS 테이블 공간의 최대 테이블	65 534
26	테이블의 의무 규정 최대수	저장영역
27	최대 레벨의 부속 조회 중첩	저장영역
28	단일 명령문의 최대 부속 조회 수	저장영역
29	INSERT문의 최대 값[g]	1 012
30	단일 UPDATE문의 대부부의 SET절[g]	1 012
31	UNIQUE 강제규정의 최대 컬럼(UNIQUE 색인을 통해 지원됨)	16
32	UNIQUE 강제규정의 최대 결합 컬럼 길이 (UNIQUE 색인을 통해 지원됨)(바이트)	1 024

표 33. 데이터베이스 관리 프로그램 한계 (계속)

	설명	한계
33	외부 키의 최대 참조 컬럼	16
34	외부 키에서의 최대 결합 참조 컬럼 길이(바이트)	1 024
35	점검 제한조건 스펙의 최대 길이(바이트)	65 535
36	파티션 키의 최대 컬럼 수 [e]	500
37	작업 단위(UOW)에서 변경된 최대 행 수	저장영역
38	최대 패키지 수	저장영역
39	명령문 내의 최대 상수	저장영역
40	서버의 최대 동시 사용자 수 [d]	64 000
41	저장된 프로시듀어의 최대 매개변수 수	32 767
42	사용자 정의 함수의 최대 매개변수 수	90
43	연쇄 트리거의 최대 런타임 길이	16
44	동시 사용 중인 이벤트 모니터의 최대 수	32
45	일반 DMS 테이블 공간의 최대 *크기* (기가바이트)[c] [g]	512
46	긴 DMS 테이블 공간의 최대 *크기*(테라바이트)[c]	2
47	임시 DMS 테이블 공간의 최대 *크기*(테라바이트)[c]	2
48	동시 사용되는 인스턴스당 최대 데이터베이스 수	256
49	인스턴스당 최대 동시 사용자 수	64 000
50	데이터베이스당 최대 동시 사용 응용프로그램 수	1 000
51	연쇄 트리거의 최대 깊이	16
52	최대 파티션 번호	999
53	DMS 테이블 공간의 대부분의 테이블 오브젝트[f]	51 000
54	가장 긴 변수 색인 키 부분(바이트 단위)	255
55	별명으로 참조되는 데이터 소스 테이블이나 뷰에서의 최대 컬럼 수	5 000
56	32비트 릴리스용 버퍼 풀의 최대 NPAGES	524 288
57	64비트 릴리스용 버퍼 풀의 최대 NPAGES	2 147 483 647
58	저장 프로시듀어에 대한 최대 중첩 레벨 수	16
59	데이터베이스내의 최대 테이블 공간 수	4096
60	구조화 유형의 최대 속성 수	4082

표 33. 데이터베이스 관리 프로그램 한계 (계속)

설명	한계

주:

a 이 최대값은 CREATE VIEW문 내의 조인을 사용하여 얻을 수 있습니다. 그러한 뷰로부터의 선택은 선택 목록 내의 대부분 요소에 대한 한계를 제시합니다.

b BLOB, CLOB, LONG VARCHAR, DBCLOB 및 LONG VARGRAPHIC 컬럼에 대한 실제 데이터는 여기에 포함되지 않습니다. 그러나, 그 데이터 위치에 관한 정보는 행에 일부 공간을 차지합니다.

c 표시되는 수는 구조적인 한계를 가지며 근사값으로 표시됩니다. 실제 한계는 이것보다 작을 수 있습니다.

d 실제 값은 MAXAGENTS 구성 매개변수의 값이 됩니다. MAXAGENTS에 대해서는 *관리 안내서*에서 정보를 참조하십시오.

e 이것은 구조적 한계입니다. 색인 키의 대다수 컬럼에 대한 한계가 실질적인 한계로 사용되어야 합니다.

f 테이블 오브젝트는 데이터, 색인, LONG VARCHAR/VARGRAPHIC 컬럼 및 LOB 컬럼을 포함합니다. 테이블 데이터와 동일한 테이블 공간에 있는 테이블 오브젝트는 한계에 여분은 포함시키지 않습니다. 단, 테이블 데이터와 다른 테이블 공간에 있는 각 테이블 오브젝트는 테이블 오브젝트가 상주하는 테이블 공간의 테이블마다 있는 각 테이블 오브젝트 유형의 한계에 1을 더합니다.

g 특정 값의 페이지 *크기*에 대해서는 표34에서 참조하십시오.

표 34. 데이터베이스 관리 프로그램 페이지 크기 특정 한계

	설명	4K 페이지 *크기* 한계	8K 페이지 *크기* 한계	16K 페이지 *크기* 한계	32K 페이지 *크기* 한계
1	테이블의 최대 컬럼	500	1 012	1 012	1 012
3	오버헤드를 모두 포함한 행의 최대 길이	4 005	8 101	16 293	32 677
4	파티션당 최대 테이블 *크기*(기가바이트)	64	128	256	512
5	파티션당 최대 색인 *크기*(기가바이트)	64	128	256	512
15	선택 목록의 최대 요소	500	1 012	1 012	1 012
17	GROUP BY절의 최대 컬럼수	500	1 012	1 012	1 012

표 34. 데이터베이스 관리 프로그램 페이지 크기 특정 한계 (계속)

	설명	4K 페이지 크기 한계	8K 페이지 크기 한계	16K 페이지 크기 한계	32K 페이지 크기 한계
18	GROUP BY절의 최대 컬럼 합계 길이(바이트)	4 005	8 101	16 293	32 677
19	ORDER BY절의 최대 컬럼 수	500	1 012	1 012	1 012
20	ORDER BY절의 최대 컬럼 합계 길이(바이트)	4 005	8 101	16 293	32 677
29	INSERT문의 최대 값	500	1 012	1 012	1 012
30	단일 UPDATE문의 최대 SET절	500	1 012	1 012	1 012
45	정규 DMS 테이블 공간의 최대 크기(기가바이트 단위)	64	128	256	512

부록B. SQL 통신(SQLCA)

SQLCA는 모든 SQL문의 실행 끝에 갱신되는 변수의 콜렉션입니다. 실행 가능한 SQL문(예외 DECLARE, INCLUDE 및 WHENEVER는 제외)을 포함하며 옵션 LANGLEVEL SAA1(기본으로)과 MIA 옵션으로 사전 처리 컴파일된 프로그램은 다중 스레드 응용프로그램에서 스레드당 한 개의 SQLCA를 가짐으로써 2개 이상의 SQLCA가 가능할 지라도 단 하나의 SQLCA만 제공해야 합니다.

프로그램이 옵션 LANGLEVEL SQL92E로 사전 처리 컴파일되면 SQLCODE 또는 SQLSTATE 변수를 SQL 선언 섹션에 선언할 수 있거나 SQLCODE 변수를 프로그램 어딘가에서 선언할 수 있습니다.

LANGLEVEL SQL92E를 사용할 때에는 SQLCA를 제공하면 안 됩니다. REXX를 제외한 모든 언어에서 SQL INCLUDE문을 SQL의 선언을 제공하는 데 사용할 수 있습니다. SQLCA는 REXX에서 자동으로 제공됩니다.

SQLCA를 대화식으로

사용자가 명령행 처리기에서 사용한 각 명령 다음에 SQLCA를 표시하려면 명령 **db2 -a**을 사용하십시오. 그러면 SQLCA가 부속 조회 명령에 대한 출력의 일부로 제공됩니다. SQLCA가 db2diag.log 파일에도 덤프됩니다.

SQLCA 필드 설명

표 35. SQLCA의 필드

이름[110]	데이터 유형	필드 값
sqlcaid	CHAR(8)	'sQLCA'를 포함하는 저장영역 덤프용 "eye catcher". SQL 프로시듀어 본문 분석시 행 번호 정보가 리턴되는 경우 6번째 바이트는 'L' 입니다.
sqlcabc	INTEGER	SQLCA의 길이(136)를 포함합니다.

110. 표시된 필드 이름은 INCLUDE문을 통해 얻은 SQLCA에 있는 필드 이름입니다.

표 35. SQLCA의 필드 (계속)

이름[110]	데이터 유형	필드 값
sqlcode	INTEGER	SQL 리턴 코드를 포함합니다. SQL 리턴 코드의 특정한 의미에 대해서는 *메시지 참조서*의 메시지 섹션을 참조하십시오. **코드** **의미** **0** 성공적인 실행(하나 이상의 SQLWARN 표시기가 설정될 수 있음). **양수** 성공적인 실행, 경고 상태 포함 **음수** 오류 상태
sqlerrml	SMALLINT	*sqlerrmc*에 대한 길이 표시기, 0에서 70 사이의 범위. 0은 *sqlerrmc의 값이* 사용되지 않았음을 의미합니다.
sqlerrmc	VARCHAR (70)	오류 상태의 설명에서 변수가 대체되고, X'FF'에 의해 분리된 하나 이상의 토큰을 포함합니다. 필드는 성공적인 연결이 완료될 때도 사용됩니다. NOT ATOMIC 복합 SQL문이 실행될 때, 최대 7개까지의 오류에 대한 정보를 포함하게 됩니다. SQL 리턴 코드의 특정한 의미에 대해서는 *메시지 참조서*의 메시지 섹션을 참조하십시오.
sqlerrp	CHAR(8)	제품의 수정 레벨과 버전 릴리스를 나타내는 4개의 디지트가 따라오며 제품을 나타내는 3개의 문자로 시작합니다. 예를 들면, SQL07010은 DB2 Universal Database 버전 7 릴리스 1 수정 레벨 0을 의미합니다. SQLCODE가 오류 상태를 나타내는 경우, 이 필드는 오류를 리턴한 모듈을 식별합니다. 필드는 성공적인 연결이 완료될 때도 사용됩니다.
sqlerrd	ARRAY	진단 정보를 제공하는 6개의 INTEGER 변수. 파티션된 데이터베이스에서 sqlerrd(6)에 대한 경우를 제외하면 오류가 없을 경우 이 값은 일반적으로 비어 있습니다.

표 35. SQLCA의 필드 (계속)

이름[110]	데이터 유형	필드 값
sqlerrd(1)	INTEGER	연결을 호출하는 데 성공한 경우, 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로 변환되면 혼합된 문자 데이터(CHAR 데이터 유형) 길이와의 최대 예상 차이를 포함합니다 0 또는 1의 값은 확장이 없음을 나타내고 1보다 큰 값은 확장이 가능함을 나타내며 음수 값은 축소가 가능함을 나타냅니다. [a] SQL 프로시듀어로부터 리턴시 SQL 프로시듀어의 리턴 상태 값을 포함합니다.
sqlerrd(2)	INTEGER	연결을 호출하는 데 성공한 경우, 데이터베이스 코드 페이지에서 응용프로그램 코드 페이지로 변환되면 혼합된 문자 데이터(CHAR 데이터 유형) 길이와의 최대 예상 차이를 포함합니다 0 또는 1의 값은 확장이 없음을 나타내고 1보다 큰 값은 확장이 가능함을 나타내며 음수 값은 축소가 가능함을 나타냅니다. SQLCA가 하나 이상의 오류를 만난 NOT ATOMIC 복합 SQL문의 결과인 경우 값은 실패한 명령문 수로 설정됩니다.
sqlerrd(3)	INTEGER	PREPARE 호출에 성공할 경우, 리턴될 행의 추정치를 포함합니다. INSERT, UPDATE 및 DELETE 다음에 영향을 받은 행의 실제 수를 포함합니다. 복합 SQL을 호출할 경우, 모든 부속 명령문 행의 누적을 포함합니다. CONNECT를 호출할 경우, 데이터베이스가 갱신되면 1을 포함하고 데이터베이스가 읽기 전용이면 2를 포함합니다. SQL 프로시듀어에 대한 CREATE PROCEDURE가 호출되고 SQL 프로시듀어 본문 분석시 오류가 발견되는 경우, 오류가 발견된 행 번호를 포함합니다. sqlcaid의 6번째 바이트는 'L'이어야 합니다.
sqlerrd(4)	INTEGER	PREPARE 호출에 성공할 경우, 명령문 처리에 필요한 자원의 비교 상대적 추정치를 포함합니다. 복합 SQL을 호출할 경우, 성공적인 부속 명령문의 수 계산을 포함합니다. CONNECT을 호출한 경우, 다운 레벨 클라이언트에서 1단계 확약에 대해 0을, 1단계 확약에 대해 1을, 1단계와 읽기 전용 확약에 대해 2를, 2단계 확약에 대해 3을 포함합니다.

표 35. SQLCA의 필드 (계속)

이름[110]	데이터 유형	필드 값
sqlerrd(5)	INTEGER	다음 두 경우의 결과로서 삭제, 삽입, 갱신된 행의 총 수를 포함합니다. • 성공적인 삭제 조작 후 제한조건의 적용 • 활성화된 트리거에서 트리거된 SQL문의 처리 복합 SQL을 호출할 경우, 모든 부속 명령문 행의 누적을 포함합니다. 일부 경우에는, 필드 내부 오류 포인터인 음수 값을 포함합니다. CONNECT를 호출할 경우, 서버 인증에 인증 유형 값 0을, 클라이언트 인증에 1을, DB2 연결을 사용한 인증에 2를, DEC 보안 서비스 인증에 3을, 지정되지 않은 인증에 255를 포함합니다.
sqlerrd(6)	INTEGER	파티션된 데이터베이스에 대해 오류 또는 경고를 만난 파티션의 파티션 번호를 포함합니다. 오류 또는 경고를 만나지 않으면, 이 필드는 조정자 노드의 파티션 번호를 포함합니다. 이 필드에 있는 번호는 db2nodes.cfg 파일의 파티션에 대해 지정된 것과 동일한 것입니다.
sqlwarn	Array	공백 또는 W를 각각 포함하는 일련의 경고 표시기. 복합 SQL문을 호출할 경우, 모든 부속 명령문에 설정한 경고 표시기의 누적을 포함합니다.
sqlwarn0	CHAR(1)	다른 모든 표시기가 공백이면 공백, 적어도 하나 다른 표시기가 공백이 아니면 W를 포함합니다.
sqlwarn1	CHAR(1)	호스트 변수에 할당될 때 문자열 컬럼의 값이 절단되면 W를 포함합니다. 널(NULL) 종료문자가 절단되면 N을 포함합니다. CONNECT 또는 ATTACH에 성공하고 연결 AuthID가 8바이트를 넘는 경우 A를 포함합니다.
sqlwarn2	CHAR(1)	널(NULL) 값이 함수의 인수에서 제거되면 W를 포함합니다.[b]
sqlwarn3	CHAR(1)	컬럼 번호가 호스트 변수의 번호와 같지 않으면 W를 포함합니다.
sqlwarn4	CHAR(1)	준비된 UPDATE 또는 DELETE문이 WHERE절을 포함하지 않으면 W를 포함합니다.
sqlwarn5	CHAR(1)	이후 사용을 위해 예약됨.
sqlwarn6	CHAR(1)	날짜 계산의 결과가 불가능한 날짜를 피하도록 조정되면 W를 포함합니다.
sqlwarn7	CHAR(1)	이후 사용을 위해 예약됨. CONNECT가 호출되어 성공한 경우 DYN_QUERY_MGMT 데이터베이스 구성 매개변수가 사용 가능하게 되면 'E'를 포함합니다.
sqlwarn8	CHAR(1)	변환된 문자가 치환 문자로 대체되면 W를 포함합니다.

표 35. SQLCA의 필드 (계속)

이름[110]	데이터 유형	필드 값
sqlwarn9	CHAR(1)	오류가 있는 산술식이 컬럼 함수 처리중에 무시되면 W를 포함합니다.
sqlwarn10	CHAR(1)	SQLCA의 필드 중 하나에서 문자 데이터 값을 변환할 때 변환 오류가 있으면 W를 포함합니다.
sqlstate	CHAR(5)	가장 최근에 실행된 SQL문의 결과를 나타내는 리턴 코드.

주:

a 세부사항에 대해서는 *응용프로그램 개발 안내서*의 『Programming in Complex Environments』장의 『Character Conversion Expansion Factor』절을 참조하십시오.

b 결과가 널(NULL) 값의 제거에 종속되지 않기 때문에 널(NULL) 값이 제거됐음에도 불구하고 SQLWARN2를 W로 설정할 수 없습니다.

오류 보고 순서

오류 보고서의 순서는 다음과 같습니다.

1. 중대한 오류는 항상 기록됩니다. 중대한 오류가 기록될 때, SQLCA에 추가되지 않습니다.
2. 중대한 오류가 발생하면, 교착 상태 오류가 다른 오류에 앞서서 취해집니다.
3. 다른 모든 오류에 대해, 첫번째 음수 SQL 코드에 대한 SQLCA가 리턴됩니다.
4. 음수 SQL 코드가 삭제되지 않으면, 첫번째 경고(즉, 양수 SQL 코드)에 대한 SQLCA가 리턴됩니다.

   DB2 Enterprise - Extended Edition에 대해, 데이터 조작 연산이 한 개의 파티션에서는 공백이지만 다른 노드에 데이터를 갖고 있는 테이블에서 실행되면 이 규칙에 대한 예외가 발생합니다. 모든 파티션의 테이블이 비어 있거나 UPDATE문의 WHERE절을 만족시키는 행이 없기 때문에 모든 파티션의 에이전트가 SQL0100W를 리턴하는 경우에는 SQLCODE + 100만이 응용프로그램으로 리턴됩니다.

DB2 Extended Enterprise Edition에서 SQLCA의 사용

DB2 Universal Database Enterprise - Extended Edition에서, 다른 파티션의 에이전트 번호에 의해 한 개의 SQL문이 수행되며 각 에이전트는 다른 오류나 경고에 대해 다른 SQLCA를 리턴합니다. 조정자 에이전트는 또한 자기 소유의 SQLCA를 갖습니다.

응용프로그램에 일관성 있는 관점을 제공하기 위해, 모든 SQLCA 값은 한 개의 구조로 병합되어야 하며 SQLCA 필드는 글로벌 계산을 나타내야 합니다. 예를 들면,

- 모든 오류와 경고에 대해, *sqlwarn*는 모든 에이전트에서 접수한 경고 플래그를 포함합니다.
- 행 계산을 나타내는 *sqlerrd* 필드에 있는 값은 모든 에이전트로부터의 누적입니다.

SQLSTATE 09000는 트리거된 SQL문 처리중 발생하는 오류의 경우에는 리턴되지 않을 수도 있음에 유의하십시오.

부록C. SQL 설명자 영역(SQLDA)

SQLDA는 SQL DESCRIBE문 실행에 필요한 변수의 콜렉션입니다. SQLDA 변수는 PREPARE, OPEN, FETCH, EXECUTE 및 CALL문에서 사용될 수 있는 옵션입니다. SQLDA는 동적 SQL과 통신하고, DESCRIBE문에서 사용되며, 호스트 변수 주소로 수정되고, FETCH문에서 재사용됩니다.

SQLDA는 모든 언어를 지원하나, 사전 정의된 선언문은 C, REXX, FORTRAN 및 COBOL에 대해서만 제공됩니다. REXX에서 SQLDA는 다른 언어에서와 약간 다릅니다. REXX에서의 SQLDA 사용에 대해 자세히 알려면 *응용프로그램 개발 안내서*를 참조하십시오.

SQLDA에 있는 정보 의미는 사용방식에 따라 다릅니다. PREPARE 및 DESCRIBE에서 SQLDA는 준비된 명령문에 관한 정보를 응용프로그램에 제공합니다. OPEN, EXECUTE, FETCH 및 CALL에서 SQLDA은 호스트 변수를 설명합니다.

DESCRIBE 및 PREPARE에서 기술된 컬럼 중 하나가 LOB 유형[111], 참조 유형 또는 사용자 정의 유형인 경우, SQLDA 전체에 대한 SQLVAR 항목 수는 두 배가 됩니다. 예를 들면,

- 3 VARCHAR 컬럼과 1 INTEGER 컬럼으로 테이블을 기술할 때 4 SQLVAR 항목이 있게 됩니다.
- 2 VARCHAR 컬럼, 1 CLOB 컬럼, 그리고 1 정수 컬럼으로 테이블을 기술할 때 8 SQLVAR 항목이 있게 됩니다.

EXECUTE, FETCH, OPEN 및 CALL에서 기술된 변수 중 하나가 LOB 유형 111 또는 구조화 유형이라면, SQLDA 전체에 대한 SQLVAR 항목 수는 두 배가 되어야 합니다.[112]

111. LOB 위치 지정자 및 파일 참조 변수에는 2배로 된 SQLDA가 필요하지 않습니다.

112. 이 경우 데이터베이스에 두 배 항목의 추가 정보가 필요하지 않으므로 구별 유형과 참조 유형은 관련되지 않습니다.

필드 설명

SQLDA는 네 변수와 SQLVAR로 명명되는 일련의 변수 어커런스 수로 이루어져 있습니다. OPEN, FETCH, EXECUTE 및 CALL에서 발생하는 각 SQLVAR은 호스트 변수를 설명합니다. DESCRIBE 및 PREPARE에서 SQLVAR의 각 어커런스는 결과 테이블의 컬럼을 기술합니다. SQLVAR 항목에는 두 종류가 있습니다.

1. **기본 SQLVAR:** 세 가지 항목이 항상 존재합니다. 여기에는 데이터 유형 코드, 길이 속성, 컬럼 이름, 호스트 변수 주소 및 표시기 변수 주소와 같은 컬럼 또는 호스트 변수에 대한 기본 정보가 포함됩니다.
2. **2차 SQLVAR:** 이들 항목은 위에 기술된 규칙에 따라 SQLVAR 항목의 수가 배로 되는 경우에만 제시됩니다. 사용자 정의 유형(구별 또는 구조화)의 경우에는 사용자 정의 유형 이름이 포함됩니다. 참조 유형의 경우에는 참조의 목표 유형이 포함됩니다. LOB의 경우에는 호스트 변수의 길이 속성과 실제 길이를 포함하는 버퍼에 대한 포인터가 포함됩니다.[113] 위치 지정자 또는 파일 참조 변수가 LOB를 나타내는 데 사용되면, 이들 항목이 필요없습니다.

두 종류의 항목이 들어 있는 SQLDA에서, 기본 SQLVAR은 2차 SQLVAR 블록 앞에 있는 블록에 있습니다. 각각의 경우, 항목 수는 SQLD의 값과 같습니다 (많은 2차 SQLVAR 항목이 사용되지 않는 경우에도).

SQLVAR 항목이 DESCRIBE에 의해 설정되는 환경은 1274 페이지의 『SQLDA에서 DESCRIBE의 영향』에서 설명합니다.

113. 구별 유형과 LOB 정보는 겹치치 않으므로, 구별 유형은 DESCRIBE의 SQLVAR 항목 수를 세배로 만들지 않고 LOB에 근거할 수 있습니다.

SQLDA 헤더 내의 필드

표 36. SQLDA 헤더 내의 필드

C 이름	SQL 데이터 유형	DESCRIBE 및 PREPARE에서의 사용 (SQLN을 제외한 데이터베이스 관리 프로그램에 의해 설정됨)	FETCH, OPEN, EXECUTE 및 CALL에서의 사용(명령문 실행 이전에 응용프로그램에서 설정)
sqldaid	CHAR(8)	이 필드의 7번째 바이트는 SQLDOUBLED라는 플래그 바이트입니다. 데이터베이스 관리 프로그램은 각 컬럼마다 두 개의 SQLVAR 항목이 작성된 경우 SQLDOUBLED를 문자 '2'로 설정합니다. 그 외에는 공백으로 설정됩니다(ASCII로는 X'20', EBCDIC으로는 X'40'). SQLDOUBLED이 설정되는 시기에 대해 1274 페이지의 『SQLDA에서 DESCRIBE의 영향』에서 자세한 내용을 참조하십시오.	이 필드의 일곱번째 바이트는 SQLVAR의 수가 두배가 될 때 사용됩니다. SQLDOUBLED가 그 이름입니다. 기술되는 호스트 변수가 구조화 유형, BLOB, CLOB 또는 DBCLOB라면 7번째 바이트를 문자 '2'로 설정해야 합니다. 그렇지 않을 경우에는 임의 문자로 설정할 수 있으나 공백 사용을 권장합니다. CALL문과 함께 사용되고 하나 이상의 SQLVAR이 데이터 필드를 FOR BIT DATA로 정의하는 경우, 6번째 바이트를 '+' 문자로 설정해야 합니다. 그렇지 않을 경우에는 임의 문자로 설정할 수 있으나 공백 사용을 권장합니다.
sqldabc	INTEGER	32 비트의 경우, SQLDA 길이는 SQLN*44+16과 같습니다. 64 비트의 경우, SQLDA 길이는 SQLN*56+16과 같습니다.	32 비트의 경우, SQLDA 길이는 >= SQLN*44+16입니다. 64 비트의 경우, SQLDA 길이는 >= SQLN*56+16입니다.
sqln	SMALLINT	데이터베이스 관리 프로그램에서 변경하지 않습니다. DESCRIBE문이 실행되기 전에 제로 이상의 값으로 설정되어야 합니다. SQLVAR 어커런스 수의 총 합계를 표시합니다.	SQLDA에 제공된 SQLDA의 총 어커런스 수. SQLN은 제로 이상의 값으로 설정되어야 합니다.
sqld	SMALLINT	데이터베이스 관리 프로그램에 의해 결과 테이블의 컬럼 수(또는 기술되는 명령문이 select문이 아닌 경우에는 제로)로 설정됩니다.	SQLVAR 어커런스로 기술되는 호스트 변수의 수.

기본 SQLVAR의 한 어커런스 내의 필드

표 37. 기본 SQLVAR 내의 필드

이름	데이터 유형	DESCRIBE 및 PREPARE에서의 사용	FETCH, OPEN, EXECUTE 및 CALL에서의 사용
sqltype	SMALLINT	컬럼 데이터 유형 및 널(NULL) 포함 여부를 표시합니다. 1276 페이지의 표39는 허용되는 값과 그 의미를 나열합니다. 구별 또는 참조 유형의 경우, 기본 유형의 데이터 유형이 이 필드에 위치함에 유의하십시오. 구조화 유형의 경우, 해당 유형에 대한 변환 그룹(CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터를 기초로 함)의 FROM SQL 변환 함수 결과에 대한 데이터 유형이 이 필드에 놓입니다. 기본 SQLVAR에서 사용자 정의 유형이나 참조 유형 설명의 일부라는 표시가 없습니다.	호스트 변수에 대해서 동일. 날짜/시간 값에 대한 호스트 변수는 문자열 변수여야 합니다. FETCH의 경우, 날짜/시간 유형 코드는 고정 길이의 문자열을 의미합니다. sqltype이 짝수 값이면, sqlind 필드가 무시됩니다.
sqllen	SMALLINT	컬럼의 길이 속성. 날짜/시간 컬럼의 경우, 값의 문자열 표시 길이. 1276 페이지의 표39에서 참조하십시오. 대형 오브젝트(LOB) 문자열에 대한 값이 0으로 설정됨에 유의하십시오(길이 속성이 2바이트 정수 정도로 작은 경우에도).	호스트 변수의 길이 속성. 1276 페이지의 표39에서 참조하십시오. CLOB, DBCLOB 및 BLOB 컬럼 값은 데이터베이스 관리 프로그램에 의해 무시됨에 유의하십시오. 2차 SQLVAR의 len.sqllonglen 필드가 대신 사용됩니다.
sqldata	포인터	문자열 SQLVAR의 경우, 컬럼이 FOR BIT DATA 속성으로 정의되어 있으면 sqldata에는 0이 포함됩니다. 컬럼이 FOR BIT DATA 속성을 갖지 않는 경우, 값은 데이터의 코드화에 따라 달라집니다. 1바이트 SBCS 코드화 데이터의 경우, sqldata에는 SBCS 코드 페이지가 포함됩니다. 혼합 DBCS 코드화 데이터의 경우, sqldata에는 복합 DBCS 코드 페이지에 연관된 SBCS 코드 페이지가 포함됩니다. 일본어나 대만어 EUC 코드화 데이터의 경우, sqldata에는 복합 EUC 코드 페이지가 포함됩니다. 다른 모든 컬럼 유형의 경우, sqldata는 정의되어 있지 않습니다.	호스트 변수의 주소를 포함합니다.(이때, 패치된 데이터는 저장됩니다.)

표 37. 기본 SQLVAR 내의 필드 (계속)

이름	데이터 유형	DESCRIBE 및 PREPARE에서의 사용	FETCH, OPEN, EXECUTE 및 CALL에서의 사용
sqlind	포인터	문자열 SQLVAR의 경우, sqlind에 복합 DBCS 코드 페이지와 연관된 DBCS 코드 페이지가 들어 있으면 sqlind에는 혼합 DBCS 코드화 데이터를 제외한 0이 포함됩니다. 다른 모든 컬럼 유형의 경우, sqlind는 정의되어 있지 않습니다.	연관 표시기 변수 주소가 있는 경우, 그 주소가 포함됩니다. 그렇지 않으면 사용되지 않습니다. sqltype이 짝수 값이면, sqlind 필드가 무시됩니다.
sqlname	VARCHAR (30)	규정화되지 않은 컬럼 이름이 포함됩니다. 시스템 생성 이름(결과 컬럼은 단일 컬럼으로부터 직접 파생되지 않았고 AS절을 사용하여 그 이름을 지정하지 않습니다)을 가진 컬럼의 경우, 13번째 바이트가 X'FF'로 설정됩니다. AS절에 의해 지정된 컬럼 이름의 경우, 이 바이트는 X'00'입니다.	CALL문과 함께 사용되어 DRDA 응용프로그램 서버(AS)에 액세스하는 경우, 다음과 같이 FOR BIT DATA 문자열을 나타내도록 sqlname을 설정할 수 있습니다. • sqlname의 길이는 8입니다. • sqlname의 처음 4바이트는 X'00000000'입니다. • sqlname의 나머지 4바이트는 예약되어 있습니다(현재는 무시). 그 외에도, sqltype는 CHAR, VARCHAR 또는 LONG VARCHAR를 나타내야 하며, sqldaid 필드의 6번째 바이트는 '+' 문자로 설정되어야 합니다. 이 기법은 서버를 액세스하기 위해 DB2 Connect를 사용할 때 OPEN 및 EXECUTE와 함께 사용될 수도 있습니다.

2차 SQLVAR의 한 어커런스 내의 필드

표 38. 2차 SQLVAR 내의 필드

이름	데이터 유형	DESCRIBE 및 PREPARE에서의 사용	FETCH, OPEN, EXECUTE 및 CALL에서의 사용
len.sqllonglen	INTEGER	BLOB, CLOB, DBCLOB 컬럼의 길이 속성.	BLOB, CLOB, DBCLOB 호스트 변수의 길이 속성. 데이터베이스 관리 프로그램은 데이터 유형에 대한 기본 SQLVAR에 있는 SQLLEN 필드를 무시합니다. 길이 속성은 BLOB 또는 CLOB에 대한 바이트 수와 DBCLOB에 대한 문자 수를 저장합니다.
reserve2	32 비트의 경우는 CHAR(3), 64 비트의 경우는 CHAR (11).	사용되지 않음.	사용되지 않음.
sqlflag4	CHAR(1)	값은 SQLVAR이 sqldatatype_name에 명명된 목표 유형으로 참조 유형을 나타내는 경우 X'01'입니다. SQLVAR이 sqldatatype_name의 사용자 정의 유형 이름으로 구조화 유형을 나타내는 경우 이 값은 X'12'입니다. 그렇지 않으면, 값은 X'00'입니다.	SQLVAR이 sqldatatype_name에 명명된 목표 유형으로 참조 유형을 나타내는 경우에는 X'01'로 설정하십시오. SQLVAR이 sqldatatype_name의 사용자 정의 유형 이름으로 구조화 유형을 나타내는 경우에는 X'12'로 설정하십시오. 그렇지 않으면, 값은 X'00'입니다.

표 38. 2차 SQLVAR 내의 필드 (계속)

이름	데이터 유형	DESCRIBE 및 PREPARE에서의 사용	FETCH, OPEN, EXECUTE 및 CALL에서의 사용
sqldatalen	포인터	사용되지 않음.	BLOB, CLOB 및 DBCLOB 호스트 변수에만 사용됨. 이 필드가 NULL이면, 실제 길이(문자 단위로)는 데이터의 시작 바로 전 4바이트에 저장되어야 하며, SQLDATA는 필드 길이의 첫 바이트로 위치 지정되어야 합니다. 이 필드가 NULL이 아니면, 여기에는 해당 기본 SQLVAR에 있는 SQLDATA 필드로부터 지시된 버퍼에 있는 데이터의 실제 길이가 *바이트 단위*로 포함되어 있는 4바이트 길이 버퍼(DBCLOB 경우도 해당)에 대한 포인터가 포함됩니다. 이 필드가 사용되는지의 여부에 관계없이 len.sqllonglen 필드를 설정해야 합니다.
sqldatatype_name	VARCHAR(27)	사용자 정의 유형 컬럼의 경우, 데이터베이스 관리 프로그램이 이를 완전한 사용자 정의 유형 이름으로 설정합니다.[1] 참조 유형의 경우, 데이터베이스 관리 프로그램은 이를 참조 목표 유형의 완전한 유형 이름으로 설정합니다.	구조화 유형의 경우, 테이블 참고에 나와 있는 형식의 완전한 사용자 정의 유형 이름으로 설정하십시오.[1]
예약됨	CHAR(3)	사용되지 않음.	사용되지 않음.

표 38. 2차 SQLVAR 내의 필드 (계속)

이름	데이터 유형	DESCRIBE 및 PREPARE 에서의 사용	FETCH, OPEN, EXECUTE 및 CALL에서의 사용

주:

1. 첫번째 8바이트에는 구별 유형의 스키마 이름이 포함됩니다.(이는 필요한 경우 오른쪽으로 확장됩니다.) 9바이트는 점(.). 10-27바이트에는 오른쪽 공백으로 확장되지 않는 유형 이름의 아래 부분이 포함됩니다.

이 필드의 주 목적이 사용자 정의 유형 이름에 대한 것이어도, 이 필드는 IBM 사전 정의 데이터 유형에 대해서도 설정되어 있음에 유의하십시오. 이러한 경우, 스키마 이름은 SYSIBM이고, 이 이름의 아래 부분은 DATATYPES 카탈로그 뷰의 TYPENAME 컬럼에 저장된 이름입니다. 예를 들면,

```
유형 이름          길이     sqldatatype_name
---------          ------   ----------------
A.B                10       A       .B
INTEGER            16       SYSIBM  .INTEGER
"Frank's".SMINT    13       Frank's .SMINT
MY."type  "        15       MY      .type
```

SQLDA에서 DESCRIBE의 영향

DESCRIBE 또는 PREPARE INTO문의 경우, 데이터베이스 관리 프로그램은 항상 결과 집합에 있는 컬럼 수에 SQLD를 설정합니다.

SQLDA내의 SQLVAR은 다음 경우에 설정됩니다.

- SQLN >= SQLD 및 LOB, 사용자 정의 유형 또는 참조 유형인 컬럼이 없을 때

  첫번째 SQLD SQLVAR 항목이 설정되고, SQLDOUBLED가 공백으로 설정될 때.

- SQLN >= 2*SQLD 및 최소한 하나의 컬럼이 LOB, 사용자 정의 유형 또는 참조 유형일 때

  두 번째로 SQLD SQLVAR 항목이 설정되고, SQLDOUBLED가 '2'로 설정될 때.

- SQLD <= SQLN < 2*SQLD 및 최소한 하나의 컬럼이 구별 유형 또는 참조 유형이지만 LOB 컬럼이나 구조화 유형 컬럼이 없을 때

첫번째 SQLD SQLVAR 항목이 설정되고, SQLDOUBLED가 공백으로 설정될 때. SQLWARN 바인드 옵션이 YES인 경우, 경고 SQLCODE +237(SQLSTATE 01594)가 발행됩니다.

SQLDA의 SQLVAR은 다음의 경우 NOT으로 설정됩니다(추가 공간과 또다른 DESCRIBE의 할당이 필요).

- SQLN < SQLD 및 LOB, 사용자 정의 유형 또는 참조 유형인 컬럼이 없을 때

  SQLVAR 항목이 설정되지 않고, SQLDOUBLED가 공백으로 설정될 때. SQLWARN 바인드 옵션이 YES인 경우, 경고 SQLCODE +236(SQLSTATE 01005)가 발행됩니다.

  성공적인 DESCRIBE를 위해 SQLD SQLVAR을 할당하십시오.

- SQLN < SQLD 및 최소한 하나의 컬럼이 구별 유형 또는 참조 유형이지만 LOB 컬럼이나 구조화 유형 컬럼이 없을 때

  SQLVAR 항목이 설정되지 않고, SQLDOUBLED가 공백으로 설정될 때. SQLWARN 바인드 옵션이 YES인 경우, 경고 SQLCODE +239(SQLSTATE 01005)가 발행됩니다.

  구별 유형의 이름과 참조 유형의 목표 유형을 포함하여 성공적인 DESCRIBE을 위해 2*SQLD SQLVAR을 할당하십시오.

- SQLN < 2*SQLD 및 최소한 하나의 컬럼이 LOB 또는 구조화 유형일 때

  SQLVAR 항목이 설정되지 않고, SQLDOUBLED가 공백으로 설정될 때. 경고 SQLCODE +238(SQLSTATE 01005)이 발행됩니다(SQLWARN 바인드 옵션의 설정과는 상관없음).

  성공적인 DESCRIBE를 위해 2*SQLD SQLVAR을 할당하십시오.

LOB 컬럼에 대해 위 목록에서 언급한 설명에는 원시 유형이 LOB 유형인 구별 유형이 포함됩니다.

BIND 또는 PREP 명령의 SQLWARN 옵션은 DESCRIBE (또는 PREPARE INTO)가 경고 SQLCODE +236, +237, +239 등을 리턴할지를 제어하는 데 사용됩니다. 사용자의 응용프로그램 코드는 항상 이러한 SQLCODE가 리턴될 수 있음을 고려하는 것이 좋습니다. 선택 목록에 LOB 또는 구조화 유형 컬럼이 있고

SQLDA에 충분한 SQLVAR이 없는 경우에는 항상 경고 SQLCODE +238이 리턴됩니다. 이것은 응용프로그램이 결과 세트의 LOB 또는 구조화 유형 컬럼으로 인해 SQLVAR의 수를 두 배로 만들어야 함을 알 수 있는 유일한 방법입니다.

구조화 유형 컬럼이 기술되고 있으나 FROM SQL 변환이 정의되지 않은 경우 (CURRENT DEFAULT TRANSFORM GROUP 특수 레지스터를 사용하여 TRANSFORM GROUP을 지정하지 않았기 때문이거나(SQLSTATE 42741) 또는 이름 그룹에 정의된 FROM SQL 변환 함수가 없기 때문(SQLSTATE 42744)), DESCRIBE는 오류를 리턴합니다. 이러한 오류는 구조화 유형 컬럼이 있는 테이블의 DESCRIBE에 대해 리턴된 것과 동일한 오류입니다.

SQLTYPE 및 SQLLEN

표39는 SQLDA의 SQLTYPE 및 SQLLEN 필드에 나타날 수 있는 값을 보여줍니다. DESCRIBE 및 PREPARE INTO에서 SQLTYPE의 짝수 값은 그 컬럼이 널(NULL)을 허용하지 않는다는 것을 의미하고, 홀수 값은 컬럼에 널(NULL)이 허용된다는 것을 의미합니다. FETCH, OPEN, EXECUTE 및 CALL에서, SQLTYPE의 짝수 값은 표시기 변수가 제공되지 않는다는 것을 의미하고, 홀수 값은 SQLIND에 표시기 변수의 주소가 포함된다는 것을 표시합니다.

표 39. DESCRIBE, FETCH, OPEN, EXECUTE 및 CALL에 대한 SQLTYPE 및 SQLLEN 변수

	DESCRIBE 및 PREPARE INTO의 경우		**FETCH, OPEN, EXECUTE 및 CALL의 경우**	
SQLTYPE	**컬럼 데이터 유형**	**SQLLEN**	**호스트 변수 데이터 유형**	**SQLLEN**
384/385	날짜 시간	10	날짜의 고정 길이 문자열 표현	호스트 변수의 길이 속성
388/389	time	8	시간의 고정 길이 문자열 표현	호스트 변수의 길이 속성
392/393	timestamp	26	시각의 고정 길이 문자열 표현	호스트 변수의 길이 속성
396/397	DATALINK	컬럼의 길이 속성	DATALINK	호스트 변수의 길이 속성
400/401	N/A	N/A	NUL 종료 그래픽 문자열	호스트 변수의 길이 속성
404/405	BLOB	0 *	BLOB	사용되지 않음. *

표 39. DESCRIBE, FETCH, OPEN, EXECUTE 및 CALL에 대한 SQLTYPE 및 SQLLEN 변수 (계속)

	DESCRIBE 및 PREPARE INTO의 경우		FETCH, OPEN, EXECUTE 및 CALL의 경우	
SQLTYPE	**컬럼 데이터 유형**	**SQLLEN**	**호스트 변수 데이터 유형**	**SQLLEN**
408/409	CLOB	0 *	CLOB	사용되지 않음. *
412/413	DBCLOB	0 *	DBCLOB	사용되지 않음. *
448/449	가변 길이 문자열	컬럼의 길이 속성	가변 길이 문자열	호스트 변수의 길이 속성
452/453	고정 길이 문자열	컬럼의 길이 속성	고정 길이 문자열	호스트 변수의 길이 속성
456/457	긴 가변 길이 문자열	컬럼의 길이 속성	긴 가변 길이 문자열	호스트 변수의 길이 속성
460/461	N/A	N/A	NUL 종료 문자열	호스트 변수의 길이 속성
464/465	가변 길이 그래픽 문자열	컬럼의 길이 속성	가변 길이 그래픽 문자열	호스트 변수의 길이 속성
468/469	고정 길이 그래픽 문자열	컬럼의 길이 속성	고정 길이 그래픽 문자열	호스트 변수의 길이 속성
472/473	긴 가변 길이 그래픽 문자열	컬럼의 길이 속성	긴 그래픽 문자열	호스트 변수의 길이 속성
480/481	부동 소수점	배정밀도의 경우 8, 단정밀도의 경우 4	부동 소수점	배정밀도의 경우 8, 단정밀도의 경우 4
484/485	팩된 십진수	바이트 1은 정밀도, 바이트 2는 스케일	팩된 십진수	바이트 1은 정밀도, 바이트 2는 스케일
492/493	대 정수	8	대 정수	8
496/497	큰 정수	4	큰 정수	4
500/501	작은 정수	2	작은 정수	2
916/917	적용 불가능	적용 불가능	BLOB 파일 참조 변수.	267
920/921	적용 불가능	적용 불가능	CLOB 파일 참조 변수.	267
924/925	적용 불가능	적용 불가능	DBCLOB 파일 참조 변수.	267
960/961	적용 불가능	적용 불가능	BLOB 위치 지정자	4
964/965	적용 불가능	적용 불가능	CLOB 위치 지정자	4

표 39. DESCRIBE, FETCH, OPEN, EXECUTE 및 CALL에 대한 SQLTYPE 및 SQLLEN 변수 (계속)

	DESCRIBE 및 PREPARE INTO의 경우		FETCH, OPEN, EXECUTE 및 CALL의 경우	
SQLTYPE	**컬럼 데이터 유형**	**SQLLEN**	**호스트 변수 데이터 유형**	**SQLLEN**
968/969	적용 불가능	적용 불가능	DBCLOB 위치 지정자	4

주:

- 2차 SQLVAR의 len.sqllonglen 필드에는 컬럼의 길이 속성이 포함됩니다.
- DB2에서의 이식성을 위해 SQLTYPE이 이전 버전으로부터 변경되었습니다. 이전 버전의 값(이전 버전 SQL 참조서 참조)이 계속 지원될 것입니다.

인식되지 않고 지원되지 않는 SQLTYPES

SQLDA의 SQLTYPE 필드에 나타나는 값은 데이터의 수신처뿐 아니라 송신처에서 사용 가능한 데이터 유형 지원 레벨에 따라 다릅니다. 이는 특히 새로운 데이터 유형이 제품에 추가될 때 중요합니다.

새로운 데이터 유형은 데이터의 수신자 또는 송신자에서 지원 또는 지원되지 않을 수도 있으며, 데이터의 송신자 또는 수신자에 의해 인식 또는 인식되지 않을 수도 있습니다. 상황에 따라, 새로운 데이터 유형이 리턴될 수도 있고, 데이터 송신자 및 수신자 모두에 부합하는 호환 데이터 유형이 리턴될 수도 있으며, 오류가 발생할 수도 있습니다.

송신자 및 수신자 모두가 호환 데이터 유형을 사용하는 데 동의하면, 다음은 발생할 맵핑을 나타냅니다. 이 맵핑은 송신자 또는 수신자 중 어느 하나라도 제공된 데이터 유형을 지원하지 않을 경우에 발생합니다. 지원되지 않는 데이터 유형은 적용업무 또는 데이터베이스 관리 프로그램으로부터 제공될 수 있습니다.

데이터 유형	호환되는 데이터 유형
BIGINT	DECIMAL(19, 0)
ROWID	VARCHAR(40) FOR BIT DATA [114]

114. ROWID는 OS/390용 DB2 Universal Database 버전 6에 의해 지원됩니다.

데이터 유형이 대체되는 SQLDA에 아무런 표시도 되지 않음을 유념하십시오.

팩된(packed) 십진수

팩된 십진수는 2진 코드화 십진수(BCD) 표의 변형으로 저장됩니다. BCD에서 각 니블(4비트)은 한 디지트의 십진수를 의미합니다. 예를 들어, 0001 0111 1001은 179를 나타냅니다. 따라서, 팩된 십진수 값을 니블별로 읽으십시오. 이 값을 바이트 단위로 저장한 후 16진수로 표기된 바이트 수를 읽어 십진수로 돌아가십시오. 예를 들어, 0001 0111 1001은 2진 표기에서 00000001 01111001이 됩니다. 이 번호를 16진으로 읽으면 0179가 됩니다.

소수점은 스케일에 의해 결정됩니다. DEC(12,5) 컬럼의 경우를 예를 들면, 맨 오른쪽 5 자리는 10진 소수점 오른쪽에 위치하게 됩니다.

부호는 숫자를 표시하는 니블(nybble) 오른쪽에 니블로 표시할 수 있습니다. 양수 또는 음수 기호가 아래와 같이 표시됩니다.

표 40. 팩된 십진수에 대한 부호 표시기

	표기		
표시	**2**진	**10**진	**16**진
양수 (+)	1100	12	C
음수 (-)	1101	13	D

요약:

1. 값을 저장하려면 *p*/2+1바이트를 할당하십시오. 여기서, *p*는 정밀도를 나타냅니다.
2. 왼쪽에서 오른쪽으로 값을 표시하기 위한 니블을 할당합니다. 숫자에 균등한 정밀도가 있는 경우, 선행 0 니블이 추가됩니다. 이 지정에는 선행(중요하지 않음) 및 후미(중요) 제로 자리수가 포함됩니다.
3. 부호 니블은 마지막 바이트의 두 번째 니블이 됩니다.

압축 십진수 변환을 수행하는 다른 방법이 있습니다. 291 페이지의 『CHAR』에서 참조하십시오.

예를 들면,

컬럼	값	바이트별로 그룹된 16진수에서의 나블
DEC(8,3)	6574.23	00 65 74 23 0C
DEC(6,2)	-334.02	00 33 40 2D
DEC(7,5)	5.2323	05 23 23 0C
DEC(5,2)	-23.5	02 35 0D

십진수에 대한 SQLLEN 필드

SQLLEN 필드에는 십진수 컬럼의 정밀도(첫번째 바이트)와 스케일(두 번째 바이트)이 포함됩니다. 이식가능한 응용프로그램을 기록하는 경우, 정밀도와 스케일 바이트가 각각 설정되어야 하거나, 짧은(단) 정수로 이 둘을 함께 설정하여야 합니다. 이렇게 하면 정수의 바이트 리버설 문제를 피할 수 있습니다.

예를 들어, C에서 다음과 같이 됩니다.

```
((char *)&(sqlda->sqlvar[i].sqllen))[0] = precision;
((char *)&(sqlda->sqlvar[i].sqllen))[1] = scale;
```

부록D. 카탈로그 뷰

데이터베이스 관리 프로그램이 시스템 카탈로그 뷰의 두 세트를 작성하고 유지보수합니다. 이 부록에는 컬럼 이름과 데이터 유형을 포함하여, 각 시스템 카탈로그 뷰의 설명이 포함되어 있습니다. 모든 시스템 카탈로그 뷰는 데이터베이스 관리 프로그램이 CREATE DATABASE 명령으로 작성될 때 작성됩니다. 카탈로그 뷰를 명시적으로 작성하거나 삭제할 수 없습니다. 시스템 카탈로그 뷰는 SQL 데이터 정의문, 환경 루틴 및 특정 유틸리티에 응답하는 정상 조작중에 갱신됩니다. 시스템 카탈로그 뷰의 데이터는 정상 SQL 조회 기능을 통하여 사용 가능합니다. 일부 특정 갱신 가능한 카탈로그 뷰의 예외 외에는 정상 SQL 데이터 조작 명령을 사용하여 시스템 카탈로그 뷰를 수정할 수 없습니다.

버전 1의 카탈로그 기본 테이블에 추가로 카탈로그 뷰도 지원됩니다. 뷰는 SYSCAT 스키마에 있고 기본값으로 모든 뷰에 대한 SELECT 특권이 PUBLIC에 부여됩니다. 응용프로그램은 기본 카탈로그 테이블보다는 이 뷰에 기록되어야 합니다. SYSCAT 스키마의 부속 집합에서 형성된 뷰의 두 번째 세트는 최적화 알고리즘이 사용한 통계 정보를 포함합니다. SYSSTAT 스키마 내의 뷰는 일부 갱신 가능 컬럼을 포함합니다.

경고: 응용프로그램으로 하여금 SYSSTAT 뷰를 사용하여 특정 컬럼을 갱신하기 하고 SYSCAT 뷰를 읽기 전용으로 하려는 의도입니다. 현재로선, SYSCAT 뷰가 읽기 전용이 아닙니다. 응용프로그램 개발자는 SYSSTAT 뷰를 사용하여 카탈로그 정보를 갱신하게만 응용프로그램을 작성하라는 경고를 받습니다. SYSCAT 뷰는 다음 버전 이주 후에 읽기 전용으로 될 것입니다.

카탈로그 뷰는 기반 카탈로그 기본 테이블보다 일치된 규약에 사용하도록 설계됩니다. 컬럼의 순서는 릴리스에 따라 바뀔 수도 있습니다. 프로그래밍 로직이 영향받지 않게 하려면, SELECT *를 사용하여 기본값으로 설정하는 대신 선택 목록의 컬럼을 항상 명시적으로 지정하십시오. 기술하는 오브젝트 유형에 따라 컬럼은 일관된 이름을 가집니다.

기술된 오브젝트	컬럼 이름
테이블	TABSCHEMA, TABNAME
색인	INDSCHEMA, INDNAME
뷰	VIEWSCHEMA, VIEWNAME
제한조건	CONSTSCHEMA, CONSTNAME
트리거	TRIGSCHEMA, TRIGNAME
패키지	PKGSCHEMA, PKGNAME
유형	TYPESCHEMA, TYPENAME, TYPEID
함수	FUNCSCHEMA, FUNCNAME, FUNCID
컬럼	COLNAME
스키마	SCHEMANAME
테이블 공간	TBSPACE
노드 그룹	NGNAME
버퍼 풀	BPNAME
이벤트 모니터	EVMONNAME
작성 시간소인	CREATE_TIME

- 『갱신 가능한 카탈로그 뷰』
- 1283 페이지의 『카탈로그 뷰에 '로드맵'』
- 1285 페이지의 『갱신 가능한 카탈로그 뷰에 '로드맵'』

갱신 가능한 카탈로그 뷰

갱신 가능 뷰에는 최적화 알고리즘이 사용한 정적인 정보가 포함되어 있습니다. 이러한 뷰에 있는 몇몇 컬럼을 변경하여 가설 데이터베이스의 성능을 조사할 수 있습니다. 오브젝트(테이블, 컬럼, 함수 또는 색인)가 나타납니다. 오브젝트를 작성한 사용자가 주어진 사용자의 갱신 가능한 카탈로그 뷰에 오브젝트에 대한 CONTROL 특권을 보유하거나 명시적 DBAM 특권을 지니고 있을 때만 이들 뷰는 SYSSTAT 스키마에 있습니다. 시스템 카탈로그 기본 테이블의 상단에 정의되어 있습니다.

처음으로 임의의 통계를 변경하기 전에 RUNSTATS 명령을 실행하라는 요구를 받으면 모든 통계는 현재 상태를 반영하게 됩니다.

카탈로그 뷰에 '로드맵'

설명	카탈로그 뷰	페이지
구조화 데이터 유형 속성	SYSCAT.ATTRIBUTES	1286
데이터베이스에 대한 권한	SYSCAT.DBAUTH	1304
노드 그룹상의 버퍼 풀 구성	SYSCAT.BUFFERPOOLS	1289
노드상의 버퍼 풀 크기	SYSCAT.BUFFERPOOLNODES	1288
유형변환 함수	SYSCAT.CASTFUNCTIONS	1290
점검 제한조건	SYSCAT.CHECKS	1291
컬럼 특권	SYSCAT.COLAUTH	1292
컬럼	SYSCAT.COLUMNS	1296
점검 제한조건이 참조한 컬럼	SYSCAT.COLCHECKS	1293
키에 사용된 컬럼	SYSCAT.KEYCOLUSE	1329
세부사항 컬럼 옵션	SYSCAT.COLOPTIONS	1295
세부사항 컬럼 통계	SYSCAT.COLDIST	1294
제한조건 종속성	SYSCAT.CONSTDEP	1301
데이터 유형	SYSCAT.DATATYPES	1302
이벤트 모니터 정의	SYSCAT.EVENTMONITORS	1306
현재 모니터된 이벤트	SYSCAT.EVENTS	1308
계층(유형, 테이블, 뷰)	SYSCAT.FULLHIERARCHIES	1309
함수 종속성	SYSCAT.FUNCDEP	1310
함수 맵핑	SYSCAT.FUNCMAPPINGS	1313
함수 맵핑 옵션	SYSCAT.FUNCMAPOPTIONS	1311
함수 맵핑 매개변수 옵션	SYSCAT.FUNCMAPPARMOPTIONS	1312
함수 매개변수	SYSCAT.FUNCPARMS	1314
계층(유형, 테이블, 뷰)	SYSCAT.HIERARCHIES	1321
색인 특권	SYSCAT.INDEXAUTH	1322
색인 컬럼	SYSCAT.INDEXCOLUSE	1323
색인 종속성	SYSCAT.INDEXDEP	1324
색인	SYSCAT.INDEXES	1325
색인 옵션	SYSCAT.INDEXOPTIONS	1328
노드 그룹 정의	SYSCAT.NODEGROUPS	1332
노드 그룹 노드	SYSCAT.NODEGROUPDEF	1331
오브젝트 맵핑	SYSCAT.NAMEMAPPINGS	1330

설명	카탈로그 뷰	페이지
패키지 종속성	SYSCAT.PACKAGEDEP	1334
패키지 특권	SYSCAT.PACKAGEAUTH	1333
패키지	SYSCAT.PACKAGES	1336
파티션 맵핑	SYSCAT.PARTITIONMAPS	1340
통과 특권	SYSCAT.PASSTHRUAUTH	1341
프로시듀어 옵션	SYSCAT.PROCOPTIONS	1345
프로시듀어 매개변수 옵션	SYSCAT.PROCPARMOPTIONS	1346
프로시듀어 매개변수	SYSCAT.PROCPARMS	1347
OS/390용 DB2 Universal Database 호환성을 제공합니다.	SYSIBM.SYSDUMMY1	1285
참조 제한조건	SYSCAT.REFERENCES	1349
원격 테이블 옵션	SYSCAT.TABOPTIONS	1366
역 데이터 유형 맵핑	SYSCAT.REVTYPEMAPPINGS	1350
스키마 특권	SYSCAT.SCHEMAAUTH	1352
스키마	SYSCAT.SCHEMATA	1353
서버 옵션	SYSCAT.SERVEROPTIONS	1354
서버 옵션 값	SYSCAT.USEROPTIONS	1372
패키지내의 명령문	SYSCAT.STATEMENTS	1356
저장 프로시듀어	SYSCAT.PROCEDURES	1342
시스템 서버	SYSCAT.SERVERS	1355
테이블 제한조건	SYSCAT.TABCONST	1359
테이블 특권	SYSCAT.TABAUTH	1357
테이블	SYSCAT.TABLES	1360
테이블 공간	SYSCAT.TABLESPACES	1364
테이블 공간 사용 특권	SYSCAT.TBSPACEAUTH	1367
트리거 종속성	SYSCAT.TRIGDEP	1368
트리거	SYSCAT.TRIGGERS	1369
유형 맵핑	SYSCAT.TYPEMAPPINGS	1370
사용자 정의 함수	SYSCAT.FUNCTIONS	1316
뷰 종속성	SYSCAT.VIEWDEP	1373
뷰	SYSCAT.TABLES	1360
	SYSCAT.VIEWS	1375
랩퍼 옵션	SYSCAT.WRAPOPTIONS	1376

설명	카탈로그 뷰	페이지
랩퍼	SYSCAT.WRAPPERS	1377

갱신 가능한 카탈로그 뷰에 '로드맵'

설명	카탈로그 뷰	페이지
컬럼	SYSSTAT.COLUMNS	1379
색인	SYSSTAT.INDEXES	1383
세부사항 컬럼 통계	SYSSTAT.COLDIST	1378
테이블	SYSSTAT.TABLES	1387
사용자 정의 함수	SYSSTAT.FUNCTIONS	1381

SYSIBM.SYSDUMMY1

하나의 행을 포함합니다. 이 뷰는 OS/390용 DB2 Universal Database과 호환성을 필요로 하는 응용프로그램에 사용 가능합니다.

표 41. SYSCAT.DUMMY1 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
IBMREQD	CHAR(1)	Y	

SYSCAT.ATTRIBUTES

사용자 정의 구조화 데이터 유형에 대해 정의된 각 속성(적용될 경우, 물려받은 속성도 포함)마다 한 행이 포함됩니다.

표 42. SYSCAT.ATTRIBUTES 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
TYPESCHEMA	VARCHAR(128)		속성을 포함하는 구조화 데이터 유형의 규정화된 이름.
TYPENAME	VARCHAR(18)		
ATTR_NAME	VARCHAR(18)		속성 이름.
ATTR_TYPESCHEMA	VARCHAR(128)		속성 유형의 규정화된 이름을 포함합니다.
ATTR_TYPENAME	VARCHAR(18)		
TARGET_TYPESCHEMA	VARCHAR(128)		속성 유형이 REFERENCE인 목표 유형의 규정화된 이름. 속성 유형이 REFERENCE가 아닌 경우 널(NULL) 값입니다.
TARGET_TYPENAME	VARCHAR(18)		
SOURCE_TYPESCHEMA	VARCHAR(128)		속성이 도입된 데이터 유형 계층에서 규정화된 데이터 유형 이름. 비 상속 속성의 경우에는, 이들 컬럼이 TYPESCHEMA 및 TYPENAME과 동일합니다.
SOURCE_TYPENAME	VARCHAR(18)		
ORDINAL	SMALLINT		제로로 시작하는 구조화 데이터 유형 정의에 있는 속성의 위치.
LENGTH	INTEGER		데이터의 최대 길이. 구별 유형에 대해 0. LENGTH 컬럼은 DECIMAL 필드에 대한 정밀도를 나타냅니다.
SCALE	SMALLINT		DECIMAL 필드에 대한 스케일; DECIMAL이 아니면 0.
CODEPAGE	SMALLINT		속성의 코드 페이지. FOR BIT DATA로 정의되지 않은 문자열 속성에 대해, 값은 데이터베이스 코드 페이지입니다. 그래픽 문자열 속성의 경우, 값은 (복합) 데이터베이스 코드 페이지가 의미하는 DBCS 코드 페이지입니다. 그렇지 않으면, 값은 0입니다.
LOGGED	CHAR(1)		유형이 LOB 또는 LOB에 기초한 구별 속성에만 적용됩니다.(그렇지 않으면 공백입니다.) Y = 속성이 로그됨 N = 속성이 로그되지 않음

표 42. SYSCAT.ATTRIBUTES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
COMPACT	CHAR(1)		유형이 LOB 또는 LOB에 기초한 구별 속성에만 적용됩니다.(그렇지 않으면 공백입니다.) Y = 속성이 저장영역에 최소 설치됨 N = 속성이 최소 설치되지 않음
DL_FEATURES	CHAR(10)		DATALINK 유형 속성에만 적용됩니다. REFERENCE 유형 속성인 경우 공백입니다. 그렇지 않으면 널(NULL)입니다. 링크 유형, 제어 모드, 복구, 언링크 속성과 같은 다양한 DATALINK 기능을 코드화합니다.

SYSCAT.BUFFERPOOLNODES

노드에서 버퍼 풀의 *크기*가 SYSCAT. BUFFERPOOLS 컬럼 NPAGES의 *기본 크기*와 다른 버퍼 풀의 각 노드에 대한 행을 포함합니다.

표 43. SYSCAT.BUFFERPOOLNODES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
BUFFERPOOLID	INTEGER		내부 버퍼 풀 식별자
NODENUM	SMALLINT		노드 번호.
NPAGES	INTEGER		이 노드에서 버퍼 풀의 페이지 수

SYSCAT.BUFFERPOOLS

모든 노드그룹의 모든 버퍼 풀에 대한 행을 포함합니다.

표 44. SYSCAT.BUFFERPOOLS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
BPNAME	VARCHAR(18)		버퍼 풀 이름
BUFFERPOOLID	INTEGER		내부 버퍼 풀 식별자
NGNAME	VARCHAR(18)	예	노드 그룹 이름 (NULL 버퍼 풀이 데이터베이스의 모든 노드에 존재하는 경우)
NPAGES	INTEGER		버퍼 풀의 페이지 수
PAGESIZE	INTEGER		이 버퍼 풀에 대한 페이지 크기
ESTORE	CHAR(1)		N = 버퍼 풀을 확장 저장영역에 사용할 수 없음 Y = 버퍼 풀을 확장 저장영역에 사용할 수 있음

SYSCAT.CASTFUNCTIONS

각 유형변환 함수에 대한 행을 포함합니다. 내장 유형변환 함수는 포함하지 않습니다.

표 45. SYSCAT.CASTFUNCTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
FROM_TYPESCHEMA	VARCHAR(128)		매개변수의 데이터 유형의 규정화된 이름.
FROM_TYPENAME	VARCHAR(18)		
TO_TYPESCHEMA	VARCHAR(128)		유형변환 후 결과의 데이터 유형의 규정화된 이름.
TO_TYPENAME	VARCHAR(18)		
FUNCSCHEMA	VARCHAR(128)		규정화된 함수 이름.
FUNCNAME	VARCHAR(18)		
SPECIFICNAME	VARCHAR(18)		함수 인스턴스 이름.
ASSIGN_FUNCTION	CHAR(1)		Y = 내재된 지정 함수 N = 지정 함수가 아님

SYSCAT.CHECKS

각 CHECK 의무 규정에 대해 한 행을 포함합니다.

표 46. SYSCAT.CHECKS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
CONSTNAME	VARCHAR(18)		점검 강제 규제 이름(테이블 내의 고유성).
DEFINER	VARCHAR(128)		점검 제한조건이 정의된 상태에서 권한 부여 ID.
TABSCHEMA	VARCHAR(128)		제한조건이 적용되는 규정화된 테이블.
TABNAME	VARCHAR(128)		
CREATE_TIME	TIMESTAMP		제한조건이 정의된 상태의 시간. 이 제한조건에 사용된 함수를 분석하는 데 사용됩니다. 제한조건 다음에 작성된 함수를 선택할 수 없습니다.
QUALIFIER	VARCHAR(128)		오브젝트 정의시의 기본 스키마의 값. 미규정화된 참조를 완료하는 데 사용됩니다.
TYPE	CHAR(1)		점검 제한조건 유형: A = GENERATED ALWAYS 컬럼에 대한 시스템 생성 점검 제한조건 C = 점검 제한조건
FUNC_PATH	VARCHAR(254)		제한조건이 작성되었을 때 사용된 현재 SQL 경로.
TEXT	CLOB(64K)		CHECK절의 텍스트.

SYSCAT.COLAUTH

특권이 권한 부여 가능한 지와 특권의 유형을 나타내는 컬럼 레벨 특권을 권한 받은 그룹 또는 사용자에 대한 하나 이상의 행을 포함합니다.

표 47. SYSCAT.COLAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
GRANTOR	VARCHAR(128)		SYSIBM 또는 특권을 권한 부여한 사용자의 권한 부여 ID.
GRANTEE	VARCHAR(128)		특권을 부여한 그룹 또는 사용자의 권한 부여 ID.
GRANTEETYPE	CHAR(1)		U = 권한 받은 사용자가 개별 사용자임 G = 권한 받은 사용자가 그룹임
TABSCHEMA	VARCHAR(128)		규정화된 테이블 또는 뷰 이름
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		이 특권이 적용되는 컬럼 이름
COLNO	SMALLINT		테이블이나 뷰의 컬럼 번호
PRIVTYPE	CHAR(1)		다음은 테이블이나 뷰에 보유된 특권의 유형을 나타냅니다. U = 갱신 특권 R = 참조 특권
GRANTABLE	CHAR(1)		특권이 권한 부여 가능한지 나타냅니다. G = 권한 부여 가능 N = 권한 부여 불가능

SYSCAT.COLCHECKS

각 행은 CHECK 의무 규정이 참조한 컬럼을 나타냅니다.

표 48. SYSCAT.COLCHECKS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
CONSTNAME	VARCHAR(18)		점검 제한조건 이름. (테이블 내의 고유성. 작성된 시스템일 수도 있습니다.)
TABSCHEMA	VARCHAR(128)		참조된 컬럼을 포함하는 규정화된 테이블 이름.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		컬럼 이름
USAGE	CHAR(1)		R = 컬럼은 점검 제한조건에서 참조됩니다. S = 컬럼은 생성된 컬럼을 지원하는 시스템 생성 점검 제한조건에서 소스 컬럼입니다. T = 컬럼은 생성된 컬럼을 지원하는 시스템 생성 점검 제한조건에서 목표 컬럼입니다.

SYSCAT.COLDIST

최적화 알고리즘의 사용을 위해 세부사항 컬럼 통계가 포함됩니다. 각 행은 컬럼의 N번째로 가장 빈번한 값을 설명합니다.

표 49. SYSCAT.COLDIST 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TABSCHEMA	VARCHAR(128)		이 항목이 적용되는 규정화 된 테이블 이름.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		이 항목이 적용되는 컬럼 이름.
TYPE	CHAR(1)		F = 빈도(자주 사용되는 값) Q = Quantile 값
SEQNO	SMALLINT		• TYPE = F인 경우, 이 컬럼의 N은 N번째의 자주 사용되는 값을 식별합니다. • TYPE = Q인 경우, 이 컬럼의 N은 N번째의 Quantile 값을 식별합니다.
COLVALUE	VARCHAR(254)	예	문자 리터럴 또는 널(NULL) 값으로서, 데이터 값.
VALCOUNT	BIGINT		• TYPE = F인 경우, VALCOUNT는 컬럼에서 COLVALUE의 발생 수입니다. • TYPE = Q인 경우, VALCOUNT는 그 값이 COLVALUE보다 작거나 같은 행의 수입니다.
DISTCOUNT	BIGINT	예	TYPE = Q인 경우, 이 컬럼은 COLVALUE보다 작거나 같은 구별 값의 수를 기록합니다(사용할 수 없는 경우 널(NULL)).

SYSCAT.COLOPTIONS

각각의 행은 컬럼 특유의 옵션 값을 포함합니다.

표 50. SYSCAT.COLOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TABSCHEMA	VARCHAR(128)		별명의 규정자.
TABNAME	VARCHAR(128)		컬럼에 대한 별명.
COLNAME	VARCHAR(128)		지역 컬럼 이름
OPTION	VARCHAR(128)		컬럼 옵션의 이름.
SETTING	VARCHAR(255)		값

SYSCAT.COLUMNS

테이블 또는 뷰에 대해 정의된 각 컬럼(적용될 경우, 물려받은 컬럼도 포함)마다 한 행이 포함됩니다. 모든 카탈로그 뷰는 SYSCAT.COLUMNS 테이블 내에 항목을 갖습니다.

표 51. SYSCAT.COLUMNS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
TABSCHEMA	VARCHAR(128)		이 컬럼을 포함하는 규정화된 테이블 또는 뷰 이름.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		컬럼 이름
COLNO	SMALLINT		0으로 시작하는 테이블이나 뷰 내의 컬럼 숫자 위치.
TYPESCHEMA	VARCHAR(128)		컬럼의 데이터 유형이 구별되는 경우, 유형의 규정화된 이름을 포함합니다. 그렇지 않으면, TYPESCHEMA는 값 SYSIBM를 포함하며 YPENAME는 컬럼의 데이터 유형을 포함합니다(예를 들어, 긴 형식 CHARACTER). 24보다 큰 *n*을 가진 FLOAT 또는 FLOAT(*n*)이 지정되면, TYPENAME이 DOUBLE로 재명명됩니다. 25보다 큰 *n*으로 FLOAT(*n*)가 지정되면, TYPENAME은 REAL로 재명명됩니다. 또한, NUMERIC도 DECIMAL로 재명명됩니다.
TYPENAME	VARCHAR(18)		
LENGTH	INTEGER		데이터의 최대 길이. 구별 유형에 대해 0. LENGTH 컬럼은 DECIMAL 필드에 대한 정밀도를 나타냅니다.
SCALE	SMALLINT		DECIMAL 필드에 대한 스케일. DECIMAL이 아니면 0.

표 51. SYSCAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
DEFAULT	VARCHAR(254)	예	테이블의 컬럼에 대한 기본값은 컬럼의 데이터 유형에 적합한 상수, 특수 레지스터 또는 변환 함수로 표현됩니다. 또한 키워드 NULL일 수도 있습니다. 값은 기본값으로 지정된 것으로부터 변경될 수 있습니다. 예를 들어, 날짜와 시간 상수는 ISO 양식으로 나타나고 변환 함수 이름은 한계가 정해진 스키마 이름과 식별자로 규정됩니다(주3 참조). DEFAULT절이 지정되지 않았거나 컬럼이 뷰 컬럼인 경우, NULL값.
NULLS	CHAR(1)		Y = 컬럼에 널(NULL) 입력 가능함 N = 컬럼에 널(NULL) 입력 불가능함 값은 표현식이나 함수에서 도출된 뷰 컬럼에 대해 N일 수 있습니다. 그러나 이러한 컬럼에서는 뷰 사용 명령문이 산술 오류에 대해 경고로 처리될 때 널(NULL)을 허용합니다. 주1 참조.
CODEPAGE	SMALLINT		컬럼 코드 페이지. FOR BIT DATA 속성을 가지고 정의되지 않은 문자열에 대해, 값은 데이터베이스 코드 페이지입니다. 그래픽 문자열 컬럼에 대해, 값은 (복합) 데이터베이스 코드 페이지가 의미하는 DBCS 코드 페이지입니다. 그렇지 않으면, 값은 0입니다.
LOGGED	CHAR(1)		유형이 LOB 또는 LOB에 기초를 둔 컬럼에만 적용됩니다.(그렇지 않으면 공백입니다.) Y= 컬럼이 로그됨 N= 컬럼이 로그되지 않음
COMPACT	CHAR(1)		유형이 LOB 또는 LOB에 기초를 둔 컬럼에만 적용됩니다.(그렇지 않으면 공백입니다.) Y = 컬럼이 저장영역에 최소 설치됨 N = 컬럼이 최소 설치되지 않음

표 51. SYSCAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
COLCARD	BIGINT		컬럼의 구별 값의 수. 통계가 수집되지 않은 경우 -1. H 테이블의 컬럼과 계승된 컬럼의 경우 -2.
HIGH2KEY	VARCHAR(254)	예	두 번째로 높은 컬럼 값. 통계가 총계되지 않은 경우와 H 테이블의 상속된 컬럼의 경우에는 이 필드가 공백입니다. 주2 참조.
LOW2KEY	VARCHAR(254)	예	두 번째로 낮은 컬럼 값. 통계가 총계되지 않은 경우와 H 테이블의 상속된 컬럼의 경우에는 이 필드가 공백입니다. 주2 참조.
AVGCOLLEN	INTEGER		평균 컬럼 길이. Long 필드 또는 LOB나 통계가 수집되지 않은 경우 -1. H 테이블의 컬럼과 계승된 컬럼의 경우 -2.
KEYSEQ	SMALLINT	예	테이블 기본 키 내의 컬럼 숫자 위치. 서브테이블 및 계층 테이블의 경우에는 이 필드가 널(NULL)입니다.
PARTKEYSEQ	SMALLINT	예	테이블 파티션 키 내의 컬럼 숫자 위치. 컬럼이 파티션 키의 부분이 아닌 경우, 이 필드는 널(NULL) 또는 0입니다. 서브테이블 및 계층 테이블의 경우에는 이 필드 역시 널(NULL)입니다.
NQUANTILES	SMALLINT		이 컬럼에 대해 SYSCAT.COLDIST에 기록된 Quantile 값의 수. 통계가 수집되지 않은 경우 -1. H 테이블의 컬럼과 계승된 컬럼의 경우 -2.
NMOSTFREQ	SMALLINT		이 컬럼에 대해 SYSCAT.COLDIST에 기록된 자주 사용되는 값의 수. 통계가 수집되지 않은 경우 -1. H 테이블의 컬럼과 계승된 컬럼의 경우 -2.
NUMNULLS	BIGINT		컬럼에 있는 널(NULL)의 갯수가 들어 있습니다. 통계를 내지 않았을 경우에는 -1.
TARGET_TYPESCHEMA	VARCHAR(128)	예	컬럼 유형이 REFERENCE인 경우, 목표 유형의 규정화된 이름. 컬럼 유형이 REFERENCE가 아닌 경우 널(NULL) 값.
TARGET_TYPENAME	VARCHAR(18)	예	
SCOPE_TABSCHEMA	VARCHAR(128)	예	컬럼 유형이 REFERENCE인 경우, 영역(목표 테이블)의 규정화된 이름. 컬럼 유형이 REFERENCE가 아니거나 영역이 지정되지 않은 경우 널(NULL) 값.
SCOPE_TABNAME	VARCHAR(128)	예	

표 51. SYSCAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
SOURCE_TABSCHEMA	VARCHAR(128)		컬럼이 도입된 각 계층에 있는 테이블 또는 뷰의 규정된 이름. 비 상속 컬럼의 경우에는, 값이 TBCREATOR 및 TBNAME과 똑같습니다. 비 유형화된 테이블 및 뷰의 컬럼에 대해서는 널(NULL)
SOURCE_TABNAME	VARCHAR(128)		
DL_FEATURES	CHAR(10)	예	DATALINK 유형 컬럼에만 적용됩니다. 그렇지 않으면 널(NULL)입니다. 각 문자 위치는 다음과 같이 정의됩니다. 1. 링크 유형(URL인 경우 U) 2. 링크 제어(파일인 경우 F, 아니오인 경우 N) 3. 무결성(모두인 경우 A, 없음인 경우 N) 4. 읽기 권한(파일 시스템인 경우 F, 데이터베이스인 경우 D) 5. 쓰기 권한(파일 시스템인 경우 F, 블록된 경우 B) 6. 복구(예인 경우 Y, 아니오인 경우 N) 7. 언링크시(복원인 경우 R, 삭제인 경우 D, 적용 불가능한 경우 N) 차후 사용을 위해 8자에서 10자가 예약되어 있습니다.
SPECIAL_PROPS	CHAR(8)	예	REFERENCE 유형 컬럼에만 적용됩니다. 그렇지 않으면 널(NULL)입니다. 각 문자 위치는 다음과 같이 정의됩니다. 오브제트 식별자(OID) 컬럼(예인 경우 Y, 아니오인 경우 N) 사용자 생성 또는 시스템 생성(사용자의 경우 U, 시스템의 경우 S)
HIDDEN	CHAR(1)		은닉 컬럼의 유형 S = 시스템 관리되는 숨겨진 컬럼 컬럼이 은닉되지 않는 경우에는 공백
INLINE_LENGTH	INTEGER		기본 테이블 행으로 보유할 수 있는 구조화 유형 컬럼의 길이. ALTER/CREATE TABLE문으로 명시적 설정된 값이 없는 경우에는 0.

표 51. SYSCAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
IDENTITY	CHAR(1)		'Y'는 컬럼이 식별 컬럼임을 나타내고 'N'은 컬럼이 식별 컬럼이 아님을 나타냅니다.
GENERATED	CHAR(1)		생성된 컬럼의 유형 A = 컬럼 값이 항상 생성됨 D = 컬럼 값이 기본값으로 생성됨 컬럼이 생성되지 않는 경우에는 공백
TEXT	CLOB(64K)		키워드 AS로 시작되는 생성된 컬럼의 텍스트를 포함합니다.
REMARKS	VARCHAR(254)	예	사용자 적용 주석.

주:

1. 버전 2로부터 시작, 값 D(기본값은 널(NULL)이 아님)는 더 이상 사용되지 않습니다. 대신에 DEFAULT 컬럼에서 널(NULL)이 아닌 값으로 WITH DEFAULT을 사용합니다.
2. 버전 2로부터 시작, 숫자 데이터의 표현이 문자 리터럴로 변경됩니다. 크기가 16에서 33바이트까지 확장됩니다.
3. 버전 2. 1.0에 대해, 변환 함수 이름은 한계가 정해지지 않으며 DEFAULT 컬럼에 이런식으로 나타나게 될 것입니다. 또한, 일부 뷰 컬럼은 DEFAULT 컬럼에 나타나게 될 기본값을 포함합니다.

SYSCAT.CONSTDEP

일부 다른 오브젝트에서 제한조건의 모든 종속성에 대한 행을 포함합니다.

표 52. SYSCAT.CONSTDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
CONSTNAME	VARCHAR(18)		제한조건 이름.
TABSCHEMA	VARCHAR(128)		제한조건이 적용되는 규정화된 테이블 이름.
TABNAME	VARCHAR(128)		
BTYPE	CHAR(1)		제한조건에 종속되는 오브젝트 유형. 가능한 값: F = 함수 인스턴스 I = 색인 인스턴스 R = 구조화 유형
BSCHEMA	VARCHAR(128)		제한조건에 종속되는 규정화된 오브젝트 이름.
BNAME	VARCHAR(18)		

SYSCAT.DATATYPES

모든 데이터 유형, 내장 및 사용자 정의 유형 포함에 대한 행을 포함합니다.

표 53. SYSCAT.DATATYPES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TYPESCHEMA	VARCHAR(128)		규정화된 데이터 유형 이름(내장 유형에 대해, TYPESCHEMA는 SYSIBM입니다).
TYPENAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		유형이 작성될 때 권한 부여 ID.
SOURCESCHEMA	VARCHAR(128)	예	구별 유형에 대해 규정화된 소스 유형 이름. 구조화 유형에 대한 참조 표시로서 사용되는 참조 유형으로서 사용되는 내장 유형의 규정된 이름. 다른 유형에 대해서는 널(NULL)입니다.
SOURCENAME	VARCHAR(18)	예	
METATYPE	CHAR(1)		시스템 사전정의 유형 T = 구별 유형 R = 구조화 유형
TYPEID	SMALLINT		데이터 유형의 시스템 생성 내부 식별자.
SOURCETYPEID	SMALLINT	예	소스 유형의 내부 ID(내장 유형에 대해 널(NULL)). 사용자 정의 구조화 유형인 경우, 이것은 참조 표시 유형의 내부 유형 ID입니다.
LENGTH	INTEGER		유형의 최대 길이. 시스템 사전 정의 매개변수 유형에 대해 0(예를 들어, DECIMAL 및 VARCHAR). 사용자 정의 구조화 유형인 경우, 이것은 참조 표시 유형의 길이를 나타냅니다.
SCALE	SMALLINT		시스템 사전정의 DECIMAL 유형에 따른 구별 유형 또는 참조 표시 유형에 대한 스케일. 모든 다른 유형에 대해 0(DECIMAL 그 자체를 포함). 사용자 정의 구조화 유형인 경우, 이것은 참조 표시 유형의 길이를 나타냅니다.
CODEPAGE	SMALLINT		문자 및 그래픽 구별 유형에 대한 코드 페이지 또는 참조 표시 유형. 그렇지 않으면 0.
CREATE_TIME	TIMESTAMP		데이터 유형의 작성 시간.
ATTRCOUNT	SMALLINT		데이터 유형에 있는 속성의 갯수.
INSTANTIABLE	CHAR(1)		Y = 유형의 인스턴스를 작성할 수 있습니다. N = 유형의 인스턴스를 작성할 수 없습니다.

표 53. SYSCAT.DATATYPES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
WITH_FUNC_ACCESS	CHAR(1)		Y = 이러한 유형의 모든 메소드는 함수 표기법을 사용하여 호출할 수 있습니다. N = 이러한 유형의 메소드는 함수 표기법을 사용하여 호출할 수 없습니다.
FINAL	CHAR(1)		Y = 사용자 정의 유형에 부속 유형이 없음 N = 사용자 정의 유형에 부속 유형이 있음
INLINE_LENGTH	INTEGER		기본 테이블 행으로 보유할 수 있는 구조화 유형 길이. CREATE TYPE문으로 명시적 설정된 값이 없는 경우에는 0.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.DBAUTH

사용자가 부여받은 데이터베이스 권한을 기록합니다.

표 54. SYSCAT.DBAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
GRANTOR	VARCHAR(128)		특권을 권한 부여받은 사용자의 권한 부여 ID 또는 시스템.
GRANTEE	VARCHAR(128)		특권을 부여한 그룹 또는 사용자의 권한 부여 ID.
GRANTEETYPE	CHAR(1)		U = 권한 받은 사용자가 개별 사용자임 G = 권한 받은 사용자가 그룹임
DBADMAUTH	CHAR(1)		다음은 권한 받은 사용자가 데이터베이스 전반에 DBADM 권한을 보유하는지 나타냅니다. Y = 권한을 보유함 N = 권한을 보유하지 않음
CREATETABAUTH	CHAR(1)		다음은 권한 받은 사용자가 데이터베이스 (CREATETAB)에서 테이블을 작성할 수 있는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음
BINDADDAUTH	CHAR(1)		다음은 권한 받은 사용자가 데이터베이스(BINDADD)에서 새 패키지를 작성할 수 있는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음
CONNECTAUTH	CHAR(1)		다음은 권한 받은 사용자를 데이터베이스(CONNECT)에 연결할 수 있는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음
NOFENCEAUTH	CHAR(1)		다음은 권한 받은 사용자가 분리 함수를 작성하도록 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음

표 54. SYSCAT.DBAUTH 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
IMPLSCHEMAAUTH	CHAR(1)		다음은 권한 받은 사용자가 데이터베이스 (IMPLICIT_SCHEMA)에서 내재적으로 스키마를 작성할 수 있는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음
LOADAUTH	CHAR(1)		다음은 권한 받은 사용자가 데이터베이스 전반에 LOAD 권한을 보유하는지 나타냅니다. Y = 권한을 보유함 N = 권한을 보유하지 않음

SYSCAT.EVENTMONITORS

정의된 모든 이벤트 모니터에 대한 행을 포함합니다.

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
EVMONNAME	VARCHAR(18)		이벤트 모니터 이름.
DEFINER	VARCHAR(128)		이벤트 모니터 식별자의 권한 부여 ID.
TARGET_TYPE	CHAR(1)		이벤트 데이터가 기록된 목표 유형. 값: F = 파일 P = 파이프
TARGET	VARCHAR(246)		이벤트 데이터가 기록된 목표 이름. 파일의 절대 경로 이름 또는 파이프의 절대 이름.
MAXFILES	INTEGER	예	이 이벤트 모니터가 이벤트 경로에 허용하는 이벤트 파일의 최대 수. 최대가 아니거나 목표 유형이 FILE이 아니면, 널(NULL).
MAXFILESIZE	INTEGER	예	이벤트 모니터가 새 파일을 작성한 후 각 이벤트 파일이 도달할 수 있는 최대 *크기*(4K 페이지 내에). 최대가 아니거나 목표 유형이 FILE이 아니면, 널(NULL).
BUFFERSIZE	INTEGER	예	파일 목표로 이벤트 모니터가 사용한 버퍼 *크기*(4K 페이지 내에). 그렇지 않으면, 널(NULL).
IO_MODE	CHAR(1)	예	파일 입출력의 모드. B = 블록화 N = 블록화하지 않음 목표 유형이 FILE이 아닐 경우에는 널(NULL).
WRITE_MODE	CHAR(1)	예	모니터는 모니터가 활성화되었을 때, 존재하는 이벤트 데이터를 처리하는 방법을 나타냅니다. 값은 다음과 같습니다. A = 추가 R = 바꾸기 목표 유형이 FILE이 아닐 경우에는 널(NULL).
AUTOSTART	CHAR(1)		데이터베이스가 시작되면, 이벤트 모니터는 자동적으로 활성화될 것입니다. Y = 예 N = 아니오

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
NODENUM	SMALLINT		이벤트 모니터가 수행되어 이벤트를 기록하는 파티션 (또는 노드)의 번호.
MONSCOPE	CHAR(1)		영역 모니터하기. L = 지역 G = 전역
REMARKS	VARCHAR(254)	예	이후 사용을 위해 예약됨.

SYSCAT.EVENTS

모니터되는 모든 이벤트에 대한 행을 포함합니다. 일반적으로 이벤트 모니터는 복수 이벤트를 모니터합니다.

표 55. SYSCAT.EVENTS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
EVMONNAME	VARCHAR(18)		이 이벤트를 모니터하는 이벤트 모니터의 번호.
TYPE	VARCHAR(18)		모니터되는 이벤트 유형. 가능한 값: DATABASE CONNECTIONS TABLES STATEMENTS TRANSACTIONS DEADLOCKS TABLESPACES
FILTER	CLOB(32K)	예	이 이벤트에 적용하는 WHERE-절의 전체 텍스트.

SYSCAT.FULLHIERARCHIES

각 행은 서브테이블과 수퍼테이블, 서브유형과 수퍼 유형 또는 서브뷰와 수퍼뷰 사이의 관계를 나타냅니다. 바로 위아래에 있는 것을 포함하여 모든 계층적 관계가 이 뷰에 포함됩니다.

표 56. SYSCAT.FULLHIERARCHIES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
METATYPE	CHAR(1)		관계 유형을 코드화합니다. R = 구조화 유형 사이 U = 유형화 테이블 사이 W = 유형화 뷰 사이
SUB_SCHEMA	VARCHAR(128)		부속 유형, 서브테이블 또는 부속 뷰의 규정 이름.
SUB_NAME	VARCHAR(128)		
SUPER_SCHEMA	VARCHAR(128)		상위 유형, 상위 테이블 또는 상위 뷰의 규정 이름.
SUPER_NAME	VARCHAR(128)		
ROOT_SCHEMA	VARCHAR(128)		계층의 루트에 있는 테이블, 뷰 또는 유형의 규정화된 이름.
ROOT_NAME	VARCHAR(128)		

SYSCAT.FUNCDEP

각 행은 일부 기타 오브젝트에 있는 함수 또는 메소드의 종속성을 나타냅니다.

표 57. SYSCAT.FUNCDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
FUNCSCHEMA	VARCHAR(128)		다른 오브젝트에 대해 종속성을 가지는 함수 또는 메소드 이름의 규정화된 이름.
FUNCNAME	VARCHAR(18)		
BTYPE	CHAR(1)		함수 또는 메소드가 종속된 오브젝트의 유형 A = 별명 F = 함수 인스턴스 또는 메소드 인스턴스 O = 테이블이나 뷰 계층에서 모든 서브테이블 또는 서브뷰에 대한 특권 종속성 R = 구조화 유형 S = 요약 테이블 T = 테이블 U = 유형화 테이블 V = 뷰 W = 유형화 뷰 X = 확장 색인
BSCHEMA	VARCHAR(128)		함수나 메소드에 의해 종속되는 오브젝트의 규정 이름 (BTYEPE='F'인 경우에는 함수의 고유 이름입니다).
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	예	BTYPE = O, S, T, U, V 또는 W인 경우, 종속 함수나 종속 메소드에 필요한 테이블 또는 뷰의 특권을 코드화합니다. 그렇지 않으면 널(NULL).

SYSCAT.FUNCMAPOPTIONS

각각의 행은 함수 맵핑 옵션 값을 포함합니다.

표 58. SYSCAT.FUNCMAPOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
FUNCTION_MAPPING	VARCHAR(18)		함수 맵핑 이름
OPTION	VARCHAR(128)		함수 맵핑 옵션의 이름.
SETTING	VARCHAR(255)		값.

SYSCAT.FUNCMAPPARMOPTIONS

각각의 행은 함수 맵핑 매개변수 옵션 값을 포함합니다.

표 59. SYSCAT.FUNCMAPPARMOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
FUNCTION_MAPPING	VARCHAR(18)		함수 맵핑의 이름.
ORDINAL	SMALLINT		매개변수의 위치.
LOCATION	CHAR(1)		L = 지역 R = 원격
OPTION	VARCHAR(128)		함수 맵핑 매개변수 옵션의 이름.
SETTING	VARCHAR(255)		값.

SYSCAT.FUNCMAPPINGS

각각의 행은 함수 맵핑을 포함합니다.

표 60. SYSCAT.FUNCMAPPINGS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
FUNCTION_MAPPING	VARCHAR(18)		함수 맵핑의 이름(시스템에 의해 생성될 수도 있습니다).
FUNCSCHEMA	VARCHAR(128)	예	함수 스키마. 시스템 내장 함수인 경우에는 널(NULL).
FUNCNAME	VARCHAR(1024)	예	지역 함수의 이름(내장 또는 사용자 정의).
FUNCID	INTEGER	예	내부적으로 지정된 식별자.
SPECIFICNAME	VARCHAR(18)	예	지역 함수 인스턴스의 이름.
DEFINER	VARCHAR(128)		이 맵핑이 작성된 권한 부여 ID.
WRAPNAME	VARCHAR(128)	예	맵핑이 적용되는 랩퍼 이름.
SERVERNAME	VARCHAR(128)	예	데이터 소스의 이름.
SERVERTYPE	VARCHAR (30)	예	맵핑이 적용되는 데이터 소스의 유형.
SERVERVERSION	VARCHAR(18)	예	맵핑이 적용되는 서버 유형의 버전.
CREATE_TIME	TIMESTAMP	예	맵핑이 작성된 시간.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.FUNCPARMS

SYSCAT.FUNCTIONS에 정의된 함수 또는 메소드의 결과나 모든 매개변수의 행을 포함합니다.

표 61. SYSCAT.FUNCPARMS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
FUNCSCHEMA	VARCHAR(128)		규정화된 함수 이름.
FUNCNAME	VARCHAR(18)		
SPECIFICNAME	VARCHAR(18)		함수 인스턴스 이름(시스템 작성일 수도 있습니다).
ROWTYPE	CHAR(1)		P = 매개변수 R = 유형변환(cast) 이전 결과 C = 유형변환(cast) 이후 결과
ORDINAL	SMALLINT		ROWTYPE = P, 함수 시그니처 내의 매개변수 숫자 위치. ROWTYPE = R이고 함수가 테이블을 리턴하는 경우, 결과 테이블내의 컬럼 숫자 위치. 그렇지 않으면 0.
PARMNAME	VARCHAR(128)		매개변수 또는 결과 컬럼 이름 또는 이름이 존재하지 않는 경우 널(NULL).
TYPESCHEMA	VARCHAR(128)		규정화된 결과 또는 매개변수의 데이터 유형 이름.
TYPENAME	VARCHAR(18)		
LENGTH	INTEGER		결과 또는 매개변수의 길이. 매개변수 또는 결과가 구별 유형인 경우 0. 주1 참조.
SCALE	SMALLINT		매개변수 또는 결과의 스케일. 매개변수 또는 결과가 구별 유형인 경우 0. 주1 참조.
CODEPAGE	SMALLINT		매개변수의 코드 페이지. 0은 FOR BIT DATA 속성으로 선언된 문자 데이터에 대한 행이거나 적절하지 않음을 나타냅니다.
CAST_FUNCID	INTEGER	예	내부 함수 ID.
AS_LOCATOR	CHAR(1)		Y = 매개변수 또는 결과가 위치 지정자 양식으로 전달됩니다. N = 위치 지정자 양식으로 전달되지 않습니다.

표 61. SYSCAT.FUNCPARMS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
TARGET_TYPESCHEMA	VARCHAR(128)		매개변수 또는 결과 유형이 REFERENCE인 경우, 목표 유형의 규정 이름. 매개변수 또는 결과 유형이 REFERENCE가 아닌 경우의 널(NULL) 값.
TARGET_TYPENAME	VARCHAR(18)		
SCOPE_TABSCHEMA	VARCHAR(128)		매개변수 또는 결과 유형이 REFERENCE인 경우, 영역의 규정 이름(목표 테이블). 매개변수 또는 결과 유형이 REFERENCE가 아니거나 영역이 정의되지 않은 경우의 널(NULL) 값.
SCOPE_TABNAME	VARCHAR(128)		
TRANSFORM_GRPNAME	VARCHAR(18)	예	구조화 유형 함수 매개변수에 대한 변환 그룹 이름.

주:

1. LENGTH 및 SCALE는 전래 함수(다른 함수를 정의함으로써 정의된 함수)에 대해 0으로 설정합니다. 왜냐하면 이것들은 이들 소스로부터 매개변수의 길이와 스케일을 전달받기 때문입니다.

SYSCAT.FUNCTIONS

각 사용자 정의 함수(스칼라, 테이블 또는 소스), 시스템 생성 메소드 또는 사용자 정의 메소드에 대한 행을 포함합니다. 내장 함수를 포함하지 않습니다.

주: 달리 명시되지 않는한 "함수"에 대한 설명은 메소드에도 적용됩니다.

표 62. SYSCAT.FUNCTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
FUNCSCHEMA	VARCHAR(128)		규정화된 함수 이름.
FUNCNAME	VARCHAR(18)		
SPECIFICNAME	VARCHAR(18)		함수 인스턴스 이름(시스템 작성일 수도 있습니다).
DEFINER	VARCHAR(128)		함수 정의자의 권한 부여 ID.
FUNCID	INTEGER		내부적으로 지정된 함수 ID.
RETURN_TYPE	SMALLINT		함수 리턴 유형의 내부 유형 코드.
ORIGIN	CHAR(1)		B = 내장 E = 사용자 정의, 외부 Q = 사용자 정의, SQL U = 사용자 정의, 소스 기반 S = 시스템 생성
TYPE	CHAR(1)		C = 컬럼 함수 R = 행 함수 S = 스칼라 함수 T = 테이블 함수
METHOD	CHAR(1)		Y = 메소드 N = 메소드가 아님
EFFECT	CHAR(2)		MU = 변환 메소드 OB = 관찰자 메소드 CN = 구성자 메소드 공백 = 시스템 생성 메소드가 아님
PARM_COUNT	SMALLINT		함수 매개변수의 번호.

표 62. SYSCAT.FUNCTIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
PARM_SIGNATURE	VARCHAR(180) FOR BIT DATA		내부 양식에서 매개변수 최대 90까지의 병합. 함수가 매개변수를 취할 경우 길이 0.
CREATE_TIME	TIMESTAMP		함수 작성의 시각. 버전 1 함수에 대해 0으로 설정.
QUALIFIER	VARCHAR(128)		오브젝트 정의시의 기본 스키마의 값.
WITH_FUNC_ACCESS	CHAR(1)		Y = 이 메소드는 함수 표기법을 사용하여 호출할 수 있습니다. N = 이 메소드는 함수 표기법을 사용하여 호출할 수 없습니다.
TYPE_PRESERVING	CHAR(1)		Y = 리턴 유형은 ″type-preserving″ 매개변수에 의해 제어됩니다. 모든 시스템 생성 변환 메소드는 유형 보존됩니다. N = 메소드의 리턴 유형으로 선언된 리턴 유형임
VARIANT	CHAR(1)		Y = 변함(결과가 다를 수 있음) N = 변하지 않음(결과는 일관됨) ORIGIN은 E가 아닐 경우 공백
SIDE_EFFECTS	CHAR(1)		E = 함수가 외부 부가작용을 갖습니다(호출 수가 중요합니다). N = 부가작용이 없습니다. ORIGIN은 E가 아닐 경우 공백
FENCED	CHAR(1)		Y = 분리됨 N = 분리되지 않음 ORIGIN은 E가 아닐 경우 공백
NULLCALL	CHAR(1)		Y = CALLED ON NULL INPUT N = RETURNS NULL ON NULL INPUT(피연산자가 널(NULL)인 경우, 함수 결과는 내재적으로 널(NULL)입니다.) ORIGIN은 E가 아닐 경우 공백
CAST_FUNCTION	CHAR(1)		Y = 유형변환 함수임 N = 유형변환 함수가 아님

표 62. SYSCAT.FUNCTIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
ASSIGN_FUNCTION	CHAR(1)		Y = 내재된 지정 함수 N = 지정 함수가 아님
SCRATCHPAD	CHAR(1)		Y = 함수에 스크래치 패드가 있음 N = 함수에 스크래치 패드가 없음 ORIGIN은 E가 아닐 경우 공백
FINAL_CALL	CHAR(1)		Y = 명령문의 끝 수행시 이 함수에 대한 최종 호출이 이루어 졌습니다. N = 최종 호출이 이루어지지 않았습니다. ORIGIN은 E가 아닐 경우 공백
PARALLELIZABLE	CHAR(1)		Y = 함수를 병렬로 실행할 수 있음 N = 함수를 병렬로 실행할 수 없음 ORIGIN은 E가 아닐 경우 공백
CONTAINS_SQL	CHAR(1)		다음은 함수 또는 메소드가 SQL을 포함하는지 지정합니다. C = CONTAINS SQL: SQL 데이터를 읽거나 수정하지 않는 SQL만 허용됩니다. N = NO SQL: SQL이 허용되지 않습니다. R = READS SQL DATA: SQL 데이터를 읽는 SQL만 허용됩니다.
DBINFO	CHAR(1)		DBINFO 매개변수가 외부 함수에 전달되었는지 나타냅니다. Y = DBINFO를 전달함 N = DBINFO를 전달하지 않음 ORIGIN은 E가 아닐 경우 공백
RESULT_COLS	SMALLINT		테이블 함수의 경우(TYPE=T) 결과 테이블의 컬럼 수를 포함합니다. 그렇지 않으면 1을 포함합니다.
LANGUAGE	CHAR(8)		함수 내용의 구현 언어. 가능한 값은 C, JAVA, OLE 또는 OLEDB입니다. ORIGIN이 E 또는 Q가 아닐 경우 공백.

표 62. SYSCAT.FUNCTIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
IMPLEMENTATION	VARCHAR(254)	예	ORIGIN = E인 경우, 이 함수를 구현하는 경로/모듈/함수를 식별합니다. ORIGIN = U이고 소스 함수가 내장 함수인 경우, 이 컬럼은 소스 함수의 이름과 시그니처를 포함합니다. 그렇지 않으면 널(NULL)입니다.
CLASS	VARCHAR(128)	예	LANGUAGE = JAVA인 경우, 이 함수를 구현하는 클래스를 식별합니다. 그렇지 않으면 널(NULL)입니다.
JAR_ID	VARCHAR(128)	예	LANGUAGE = JAVA인 경우, 이 함수를 구현하는 jar 파일을 식별합니다. 그렇지 않으면 널(NULL)입니다.
PARM_STYLE	CHAR(8)		CREATE FUNCTION문에 선언된 매개변수 스타일을 나타냅니다. 값: DB2SQL DB2GENRL JAVA ORIGIN은 E가 아닐 경우 공백
SOURCE_SCHEMA	VARCHAR(128)	예	ORIGIN = U이고 소스 함수가 사용자 정의 함수(UDF)인 경우, 소스 함수의 규정된 이름을 포함합니다. ORIGIN = U이고 소스 함수가 내장 함수인 경우, SOURCE_SCHEMA는 'sYSIBM'이고 SOURCE_SPECIFIC는 '내장용 N/A'입니다. ORIGIN은 U가 아닐 경우 공백
SOURCE_SPECIFIC	VARCHAR(18)	예	
IOS_PER_INVOC	DOUBLE		호출당 입출력의 평가 수. 알지 못하면 -1 (0 기본값).
INSTS_PER_INVOC	DOUBLE		호출당 명령의 평가 수. 알지 못하면 -1 (450 기본값).
IOS_PER_ARGBYTE	DOUBLE		입력 인수 바이트당 입출력의 평가 수. 알지 못하면 -1(0 기본값).
INSTS_PER_ARGBYTE	DOUBLE		입력 인수 바이트당 명령의 평가 수. 알지 못하면 -1(0 기본값).
PERCENT_ARGBYTES	SMALLINT		함수가 실제로 읽는 입력 인수 바이트의 평가 평균 퍼센트. 알지 못하면 -1(100 기본값).

표 62. SYSCAT.FUNCTIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
INITIAL_IOS	DOUBLE		처음/마지막으로 호출된 함수를 수행한 입출력의 평가 수. 알지 못하면 -1(0 기본값).
INITIAL_INSTS	DOUBLE		처음/마지막으로 호출된 함수를 실행한 명령의 평가 수. 알지못하면 -1 (0 기본값).
CARDINALITY	BIGINT		테이블 함수의 예측된 기본 행 수. 알려져 있지 않거나 함수가 테이블 함수가 아닌 경우에는 -1.
IMPLEMENTED	CHAR(1)		Y = 함수가 구현됩니다. M = 메소드가 구현되고 함수 액세스 권한을 가지지 않습니다. 주 1 참조. H = 메소드가 구현되고 함수 액세스 권한을 가집니다. 주 1 참조. N = 구현없는 메소드 방법(일반적인 방법을 지칭하는 경우).
SELECTIVITY	DOUBLE		사용자 정의 술어에 사용됨. 사용자 술어가 없을 경우에는 -1입니다. 주 2 참조.
OVERRIDEN_FUNCID	INTEGER	예	이후 사용을 위해 예약됨.
SUBJECT_TYPESCHEMA	VARCHAR(128)	예	사용자 정의 메소드를 위한 주제 유형 스키마.
SUBJECT_TYPENAME	VARCHAR(18)	예	사용자 정의 메소드를 위한 주제 유형 이름.
FUNC_PATH	VARCHAR(254)	예	함수가 정의되었을 때의 함수 경로.
BODY	CLOB(1M)	예	언어가 SQL인 경우, CREATE FUNCTION 또는 CREATE METHOD문의 텍스트.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

주:

1. 이 값은 DB2의 미래 버전에는 나타나지 않을 것입니다.
2. 이 컬럼은 모든 사용자 정의 함수에 대한 팩화된 설명자와 시스템 카탈로그에서 이주 중 -1로 설정됩니다. 사용자 정의 술어의 경우 시스템 카탈로그에서의 선택성은 -1이 됩니다. 이 경우 최적화 알고리즘에서 사용하는 선택성 값은 0.01.입니다.

SYSCAT.HIERARCHIES

각 행은 서브테이블과 바로 위의 수퍼테이블, 서브유형과 바로 위의 수퍼 유형 또는 서브뷰와 바로 위의 수퍼뷰 사이의 관계를 나타냅니다. 바로 위아래의 계층 관계만 이 뷰에 포함됩니다.

표 63. SYSCAT.HIERARCHIES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
METATYPE	CHAR(1)		관계 유형을 코드화합니다. R = 구조화 유형 사이 U = 유형화 테이블 사이 W = 유형화 뷰 사이
SUB_SCHEMA	VARCHAR(128)		부속 유형, 서브테이블 또는 부속 뷰의 규정 이름.
SUB_NAME	VARCHAR(128)		
SUPER_SCHEMA	VARCHAR(128)		상위 유형, 상위 테이블 또는 상위 뷰의 규정 이름.
SUPER_NAME	VARCHAR(128)		
ROOT_SCHEMA	VARCHAR(128)		계층의 루트에 있는 테이블, 뷰 또는 유형의 규정화된 이름.
ROOT_NAME	VARCHAR(128)		

SYSCAT.INDEXAUTH

색인에 부여된 특권에 대한 행을 포함합니다.

표 64. SYSCAT.INDEXAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
GRANTOR	VARCHAR(128)		특권을 부여받은 사용자의 권한 부여 ID.
GRANTEE	VARCHAR(128)		특권을 부여한 그룹 또는 사용자의 권한 부여 ID.
GRANTEETYPE	CHAR(1)		U = 권한 받은 사용자가 개별 사용자임 G = 권한 받은 사용자가 그룹임
INDSCHEMA	VARCHAR(128)		색인 이름.
INDNAME	VARCHAR(18)		
CONTROLAUTH	CHAR(1)		다음은 권한 받은 사용자가 색인 전반에 CONTROL 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음

SYSCAT.INDEXCOLUSE

색인에 참여하는 모든 컬럼을 나열합니다.

표 65. SYSCAT.INDEXCOLUSE 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
INDSCHEMA	VARCHAR(128)		규정화된 색인 이름.
INDNAME	VARCHAR(18)		
COLNAME	VARCHAR(128)		컬럼 이름.
COLSEQ	SMALLINT		색인에서 컬럼의 숫자 위치(초기 위치 = 1).
COLORDER	CHAR(1)		색인에서 이 컬럼에 있는 값들의 순서. 값: A = 오름차순 D = 내림차순 I = INCLUDE 컬럼(순서 무시)

SYSCAT.INDEXDEP

각 행은 일부 기타 오브젝트에 있는 색인의 종속성을 나타냅니다.

표 66. SYSCAT.INDEXDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
INDSCHEMA	VARCHAR(128)		다른 오브젝트에 대해 종속성을 가지는 색인의 규정화된 이름.
INDNAME	VARCHAR(18)		
BTYPE	CHAR(1)		색인이 종속된 오브젝트의 유형. A = 별명 F = 함수 인스턴스 O = 테이블이나 뷰 계층에서 모든 서브테이블 또는 서브뷰에 대한 특권 종속성 R = 구조화 유형 S = 요약 테이블 T = 테이블 U = 유형화 테이블 V = 뷰 W = 유형화 뷰 X = 확장 색인
BSCHEMA	VARCHAR(128)		색인이 종속성을 가지는 오브젝트의 규정화된 이름.
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	예	BTYPE = O, S, T, U, V 또는 W인 경우, 종속 색인에 필요한 테이블 또는 뷰의 특권을 코드화합니다. 그렇지 않으면 널(NULL).

SYSCAT.INDEXES

테이블에 대해 정의된 각 색인(적용될 경우, 물려받은 색인도 포함)마다 한 행이 포함됩니다.

표 67. SYSCAT.INDEXES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
INDSCHEMA	VARCHAR(128)		색인 이름.
INDNAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		색인을 작성한 사용자.
TABSCHEMA	VARCHAR(128)		색인이 정의되는 테이블이나 별명의 규정화된 이름.
TABNAME	VARCHAR(128)		
COLNAMES	VARCHAR (640)		오름차순 또는 내림차순을 나타내기 위해 각각의 앞에 + 또는 -가 오는 컬럼 이름 목록. 경고: 이 컬럼은 나중에 제거될 것입니다. 이 정보에 대해서는 1323 페이지의 『SYSCAT.INDEXCOLUSE』를 참고하십시오.
UNIQUERULE	CHAR(1)		고유 규칙: D = 중복 허용 P = 1차 색인 U = 허용된 고유 항목만
MADE_UNIQUE	CHAR(1)		Y = 색인이 원래 고유하지 않으나 고유 또는 기본 키 제한조건을 지원하기 위해 고유 색인으로 변환됩니다. 제한조건이 삭제될 경우 색인은 다시 고유하지 않은 상태로 될 것입니다. N = 색인이 작성된 대로 유지됨
COLCOUNT	SMALLINT		키 컬럼 수 + 포함 컬럼 수(있을 경우).
UNIQUE_COLCOUNT	SMALLINT		고유 키에 필요한 컬럼 수. 항상 <=COLCOUNT. 포함 컬럼이 있을 경우에만 < COLCOUNT입니다. 색인에 고유 키(중복 허용)가 없을 경우에는 -1입니다.
INDEXTYPE	CHAR(4)		색인 유형. CLUS = 클러스터링 REG = 정규

표 67. SYSCAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
ENTRYTYPE	CHAR(1)		H = 계층 테이블의 색인(H 테이블) L = 유형화 테이블의 논리 색인 유형화되지 않은 테이블상의 색인인 경우에는 공백
PCTFREE	SMALLINT		처음 색인을 구축하는 동안 예약될 각 색인 리프 페이지의 비율. 이 공간은 색인이 만들어진 후에 삽입될 경우를 대비하여 제공됩니다.
IID	SMALLINT		내부 색인 ID.
NLEAF	INTEGER		리프 페이지의 수; 통계를 내지 않았을 경우에는 -1
NLEVELS	SMALLINT		색인 레벨 수; 통계를 내지 않았을 경우에는 -1
FIRSTKEYCARD	BIGINT		구별되는 첫번째 키 값의 수. 통계를 내지 않았을 경우에는 -1.
FIRST2KEYCARD	BIGINT		색인의 처음 두 컬럼을 사용하는 구별 키의 수(통계를 내지 않았거나 적용 불가능한 경우에는 -1)
FIRST3KEYCARD	BIGINT		색인의 처음 세 컬럼을 사용하는 구별 키의 수(통계를 내지 않았거나 적용 불가능한 경우에는 -1)
FIRST4KEYCARD	BIGINT		색인의 처음 네 컬럼을 사용하는 구별 키의 수(통계를 내지 않았거나 적용 불가능한 경우에는 -1)
FULLKEYCARD	BIGINT		구별되는 전체 키 값의 수. 통계를 내지 않았을 경우에는 -1.
CLUSTERRATIO	SMALLINT		색인의 데이터 클러스터링 등급. 통계를 내지 않았거나 세부사항 색인 통계를 낸 경우에는 -1(이 경우에는 CLUSTERFACTOR가 대신 사용됩니다).
CLUSTERFACTOR	DOUBLE		클러스터링 수준의 더 정교한 측정 또는 상세 색인 통계가 수집되지 않았거나 색인이 별명에서 정의되는 경우에는 -1.
SEQUENTIAL_PAGES	INTEGER		간격 사이에 큰 간격이 없거나 거의 없을 경우 색인 키 순서의 디스크에 지정된 리프 페이지의 수(사용 가능한 통계가 없을 경우에는 -1).
DENSITY	INTEGER		색인이 차지한 페이지 범위에서 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로서, 퍼센트로 표시됩니다(0에서 100까지의 정수로 표시되고, 사용 가능한 통계가 없을 경우에는 -1임).
USER_DEFINED	SMALLINT		이 색인이 사용자에 의해 정의되었고 삭제되지 않은 경우, 1입니다.

표 67. SYSCAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
SYSTEM_REQUIRED	SMALLINT		기본 키 또는 고유 키 제한조건에 이 색인이 필요하거나 이 색인이 입력된 테이블의 오브젝트 식별자(OID) 컬럼에 있는 색인인 경우에는 1입니다. 기본 키 또는 고유 키 제한조건에 이 색인이 필요하고 이 색인이 입력된 테이블의 오브젝트 식별자(OID) 컬럼에 있는 색인인 경우에는 2입니다. 그렇지 않으면 0.
CREATE_TIME	TIMESTAMP		색인이 작성된 시간.
STATS_TIME	TIMESTAMP	예	이 색인에 대한 기록된 통계를 마지막으로 변경할 시간. 통계가 사용 가능하지 않으면 널(NULL).
PAGE_FETCH_PAIRS	VARCHAR(254)		문자 양식에서 표현된, 쌍으로 된 정수의 목록. 각 쌍은 가설 버퍼에 있는 페이지 수를 나타내며 페이지 패치의 수는 가설 버퍼를 사용하여 이 색인으로 테이블을 스캔하는 데 필요합니다. (사용 가능한 데이터가 없을 경우 문자열 길이 0)
MINPCTUSED	SMALLINT		0이 아니라면, 온라인 색인 재구성이 작동 가능화되며 값은 페이지 병합 전의 최소 사용 공간의 임계값입니다.
REVERSE_SCANS	CHAR(1)		Y = 색인이 역 스캔을 지원함 N = 색인이 역 스캔을 지원하지 않음
INTERNAL_FORMAT	SMALLINT		색인의 내부 표현을 코드화합니다.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.INDEXOPTIONS

각 행은 색인 특유의 옵션 값을 포함합니다.

표 68. SYSCAT.INDEXOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
INDSCHEMA	VARCHAR(128)		색인의 스키마 이름.
INDNAME	VARCHAR(18)		색인의 지역 이름.
OPTION	VARCHAR(128)		색인 옵션의 이름.
SETTING	VARCHAR(255)		값.

SYSCAT.KEYCOLUSE

고유 키, 기본 키 또는 외부 키 제한조건으로 정의된 키에 관계된 모든 컬럼을 나열합니다(적용될 경우, 물려받은 기본 키 또는 고유 키도 포함).

표 69. SYSCAT.KEYCOLUSE 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
CONSTNAME	VARCHAR(18)		점검 제한조건 이름. (테이블 내의 고유성)
TABSCHEMA	VARCHAR(128)		이 컬럼을 포함하는 규정화된 테이블 이름.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		컬럼 이름.
COLSEQ	SMALLINT		키에서 컬럼의 숫자 위치 (초기 위치=1).

SYSCAT.NAMEMAPPINGS

각 행은 논리적 오브젝트와 논리적 오브젝트를 구현하는 해당 구현 오브젝트간의 맵핑을 나타냅니다.

표 70. SYSCAT.NAMEMAPPINGS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TYPE	CHAR(1)		C = 컬럼 I = 색인 U = 유형화 테이블
LOGICAL_SCHEMA	VARCHAR(128)		규정된 논리 오브젝트 이름.
LOGICAL_NAME	VARCHAR(128)		
LOGICAL_COLNAME	VARCHAR(128)	예	TYPE = C인 경우 논리 컬럼의 이름. 그렇지 않으면 널(NULL).
IMPL_SCHEMA	VARCHAR(128)		논리적 오브젝트를 구현하는 구현 오브젝트의 규정화된 이름.
IMPL_NAME	VARCHAR(128)		
IMPL_COLNAME	VARCHAR(128)	예	TYPE = C인 경우 구현 컬럼의 이름. 그렇지 않으면 널(NULL).

SYSCAT.NODEGROUPDEF

노드 그룹에 포함되는 각 파티션에 대한 행을 포함합니다.

표 71. SYSCAT.NODEGROUPDEF 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
NGNAME	VARCHAR(18)		파티션(또는 노드)을 포함하는 노드 그룹 이름.
NODENUM	SMALLINT		노드 그룹에 포함되어 있는 파티션의 파티션(또는 노드) 수. 유효한 파티션 수는 0 이상 99 이하입니다.
IN_USE	CHAR(1)		파티션(또는 노드)의 상태. A = 새로 추가된 파티션이 파티션 맵에 없으나 노드 그룹의 테이블 공간에 대한 컨테이너가 작성됩니다. 재분배 노드 그룹 조작이 성공적으로 완료될 때 파티션은 파티션 맵핑에 추가됩니다. D = 노드 그룹 재분배 조작 완료시 파티션이 삭제됩니다. T = 새로 추가된 파티션이 파티션 맵에 없으며 WITHOUT TABLESPACES절을 사용하여 추가되었습니다. 컨테이너는 노드 그룹에 대한 테이블 공간에 명확하게 추가되어야 합니다. Y = 파티션이 피티션 맵에 있음

SYSCAT.NODEGROUPS

각 노드 그룹에 대한 행을 포함합니다.

표 72. SYSCAT.NODEGROUPS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
NGNAME	VARCHAR(18)		노드 그룹 이름.
DEFINER	VARCHAR(128)		노드 그룹 정의자의 권한 부여ID.
PMAP_ID	SMALLINT		SYSCAT.PARTITIONMAPS 내의 파티션 식별자.
REBALANCE_PMAP_ID	SMALLINT		현재 재분배에 사용되고 있는 파티션 맵 식별자. 재분배가 현재 처리 중이 아니면 값은 -1.
CREATE_TIME	TIMESTAMP		노드 그룹의 작성 시간.
REMARKS	VARCHAR(254)	예	사용자 제공 주석.

SYSCAT.PACKAGEAUTH

패키지에 부여된 모든 특권에 대한 행을 포함합니다.

표 73. SYSCAT.PACKAGEAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
GRANTOR	VARCHAR(128)		특권을 부여받은 사용자의 권한 부여 ID.
GRANTEE	VARCHAR(128)		특권을 부여한 그룹 또는 사용자의 권한 부여 ID.
GRANTEETYPE	CHAR(1)		U = 권한 받은 사용자가 개별 사용자임 G = 권한 받은 사용자가 그룹임
PKGSCHEMA	VARCHAR(128)		특권을 부여받은 패키지 이름.
PKGNAME	CHAR(8)		
CONTROLAUTH	CHAR(1)		다음은 권한 받은 사용자가 패키지에 대한 CONTROL 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음
BINDAUTH	CHAR(1)		다음은 권한 받은 사용자가 패키지에 대한 BIND 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음
EXECUTEAUTH	CHAR(1)		다음은 권한 받은 사용자가 패키지에 대한 EXECUTE 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음

SYSCAT.PACKAGEDEP

패키지가 색인, 테이블, 뷰, 함수, 별명, 유형, 계층에 대해 갖는 각 종속성에 대한 행이 들어 있습니다.

표 74. SYSCAT.PACKAGEDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
PKGSCHEMA	VARCHAR(128)		패키지 이름.
PKGNAME	CHAR(8)		
BINDER	VARCHAR(128)	예	패키지의 바인더.
BTYPE	CHAR(1)		오브젝트 BNAME의 유형: A = 별명 D = 서버 정의 F = 함수 인스턴스 I = 색인 M = 함수 맵핑 N = 별명 O = 테이블이나 뷰 계층에서 모든 서브테이블 또는 서브뷰에 대한 특권 종속성 P = 페이지 크기 R = 구조화 유형 S = 요약 테이블 T = 테이블 U = 유형화 테이블 V = 뷰 W = 유형화 뷰
BSCHEMA	VARCHAR(128)		패키지가 종속되는 규정화된 오브젝트 이름.
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	예	BTYPE이 O, S, T, U, V 또는 W인 경우에는 이 패키지에 필요한 특권들(선택, 삽입, 삭제, 갱신)을 코드화 합니다.

표 74. SYSCAT.PACKAGEDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명

주: 종속 함수 인스턴스가 삭제될 때, 패키지는 명시적으로 리바운드되어야 하는『사용 불가능』상태로 배치됩니다. 다른 종속 오브젝트가 삭제될 때, 패키지는『유효하지 않은』상태로 배치되어 첫번째로 참조될 경우 시스템이 자동적으로 패키지를 리바운드하려고 합니다.

SYSCAT.PACKAGES

응용프로그램프로그램을 바인드함으로써 작성된 각 패키지에 대한 행을 포함합니다.

표 75. SYSCAT.PACKAGES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
PKGSCHEMA	VARCHAR(128)		패키지 이름.
PKGNAME	CHAR(8)		
BOUNDBY	VARCHAR(128)		패키지 바인더의 권한 부여 ID(OWNER).
DEFINER	VARCHAR(128)		바운드된 패키지에 따른 사용자 ID.
DEFAULT_SCHEMA	VARCHAR(128)		정적 SQL문의 규정 이름에 사용된 기본 스키마 이름 (QUALIFIER).
VALID	CHAR(1)		Y = 유효함 N = 유효하지 않음 X = 종속되는 일부 함수 인스턴스가 삭제되었기 때문에 패키지는 작동 불능 상태입니다. 명시적 리바인드가 필요합니다. 1334 페이지의 『SYSCAT.PACKAGEDEP』에서 주 1을 참조하십시오.
UNIQUE_ID	CHAR(8)		패키지가 첫번째로 작성되었을 때 나타나는 내부 데이터 및 시간 정보.
TOTAL_SECT	SMALLINT		패키지의 총 섹션 수.
FORMAT	CHAR(1)		날짜 및 시간 형식은 다음 패키지와 연관되어 있습니다. 0 = 데이터베이스의 국가 코드와 연관된 형식 1 = USA 날짜 및 시간 2 = EUR 날짜, EUR 시간 3 = ISO 날짜, ISO 시간 4 = JIS 날짜, JIS 시간 5 = LOCAL 날짜, LOCAL 시간

표 75. SYSCAT.PACKAGES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
ISOLATION	CHAR(2)	예	분리 레벨: RR = 반복 읽기(RR) RS = 읽기 안정성(RS) CS = 커서 안정성(CS) UR = 미확약 읽기(UR)
BLOCKING	CHAR(1)	예	커서 블로킹 옵션: N = 블로킹 없음 U = 명확한 블록화 커서 B = 전체 블록화 커서
INSERT_BUF	CHAR(1)		바인드 중 사용된 삽입 옵션: Y = 삽입을 버퍼함 N = 삽입을 버퍼하지 않음
LANG_LEVEL	CHAR(1)	예	BIND 중 사용된 LANGLEVEL 값: 0 = SAA1 1 = SQL92E 또는 MIA
FUNC_PATH	VARCHAR(254)		이 패키지에 대한 마지막 BIND 명령에 사용되는 SQL 경로. 이것을 REBIND에 대해 기본값으로 사용합니다. 이전 버전 2 패키지에 대한 SYSIBM.
QUERYOPT	INTEGER		패키지를 바운드함에 따른 최적화 클래스. 재바인드를 위해 사용됨. 0, 1, 3, 5 및 9 클래스가 있습니다.
EXPLAIN_LEVEL	CHAR(1)		설명이 EXPLAIN 또는 EXPLSNAP 바인드 옵션 사용에 요청되는지 나타냅니다. P = 플랜 선택 레벨 '아니오' Explain이 요청된 경우에는 공백
EXPLAIN_MODE	CHAR(1)		EXPLAIN 바인드 옵션의 값: Y = 예(정적) N = 아니오 A = 모두(정적 및 동적)

표 75. SYSCAT.PACKAGES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
EXPLAIN_SNAPSHOT	CHAR(1)		EXPLSNAP 바인드 옵션의 값: Y = 예(정적) N = 아니오 A = 모두(정적 및 동적)
SQLWARN	CHAR(1)		동적 SQL문으로 인한 양수 SQLCODE가 응용프로그램으로 리턴됩니까? Y = 예 N = 아니오, 제한되어 있습니다.
SQLMATHWARN	CHAR(1)		바인드시 데이터베이스 구성 매개변수 DFT_SQLMATHWARN의 값. 정적 SQL문의 산술 오류와 검색 변환 오류가 경고에서 널(NULL)로 처리됩니까? Y = 예 N = 아니오, 제한되어 있습니다.
EXPLICIT_BIND_TIME	TIMESTAMP		패키지가 명시적으로 마지막 바인드되거나 리바인드된 시간. 패키지가 내재적으로 리바인드되면, 이번 보다 나중에 작성된 것으로 함수 인스턴스를 선택할 수 없습니다.
LAST_BIND_TIME	TIMESTAMP		패키지가 명시적 또는 내재적으로 마지막 바인드되거나 리바인드된 시간.
CODEPAGE	SMALLINT		바인드시 응용프로그램코드 페이지(알 수 없으면 -1).
DEGREE	CHAR(5)		패키지가 바인드될 때 파티션 내 병렬 처리(바인드 옵션처럼)에 대한 한계를 나타냅니다. 1 = 파티션 내 병렬 처리가 아님 2 - 32 767 = 파티션 내 병렬 처리의 등급 ANY = 등급이 데이터베이스 관리 프로그램에 의해 결정됨
MULTINODE_PLANS	CHAR(1)		Y = 패키지가 다중 파티션 환경에서 바인드되었음 N = 패키지가 단일 파티션 환경에서 바인드되었음

표 75. SYSCAT.PACKAGES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
INTRA_PARALLEL	CHAR(1)		패키지에서 정적 SQL으로 파티션 내 병렬 처리의 사용을 나타냅니다. Y = 패키지에서 하나 이상의 정적 SQL문이 파티션 내 병렬 처리를 사용합니다. N = 패키지에서 파티션 내 병렬 처리를 사용하는 정적 SQL문이 없습니다. F = 패키지에서 하나 이상의 정적 SQL문이 파티션 내 병렬 처리를 사용할 수 있습니다. 이러한 병렬 처리는 파티션 내 병렬 처리용으로 구성되지 않은 시스템에서는 사용할 수 없게 되어 있습니다.
VALIDATE	CHAR(1)		B = 모든 점검이 BIND중 수행되어야 함 R = 예약
DYNAMICRULES	CHAR(1)		B = 동적 SQL문이 수행시 정적 SQL문처럼 처리됩니다. 바인더 AuthID가 사용됩니다. R = 동적 SQL문이 수행시 동적 SQL문처럼 처리됩니다. 실행자 AuthID가 사용됩니다. 초기 값은 R입니다.
SQLERROR	CHAR(1)		패키지를 바인드 또는 리바인드한 가장 최근 부속 명령에 대한 SQLERROR 옵션을 나타냅니다. C = 예약 N = 패키지 없음
REFRESHAGE	DECIMAL (20,6)		요약 테이블에 대해 REFRESH TABLE문이 수행된 때와 기본 테이블 대신 요약 테이블이 사용된 때 사이의 최대 시간 길이를 나타내는 시간소인 지속시간.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.PARTITIONMAPS

테이블 파티션 키 해싱에 따라, 노드 그룹의 파티션들 사이에 테이블의 행을 분배하는 데 사용하는 각 파티션 맵에 대한 행을 포함합니다.

표 76. SYSCAT.PARTITIONMAPS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
PMAP_ID	SMALLINT		파티션 맵핑의 식별자.
PARTITIONMAP	LONG VARCHAR FOR BIT DATA		실제 파티션 맵, 여러 노드 노드 그룹에 대한 2바이트 정수 4 096의 벡터. 단일 노드 노드 그룹에 대해, 단일 노드의 파티션(또는 노드) 수를 나타내는 항목이 하나 있습니다.

SYSCAT.PASSTHRUAUTH

이 카탈로그 뷰는 통과 세션에서 데이터 소스를 조회하기 위해 권한 부여에 관한 정보를 포함합니다. 기본 테이블상의 제한조건은 SERVER에 있는 값들이 SYSCAT.SERVERS의 SERVER 컬럼에 있는 값에 해당할 것을 요구합니다. SYSCAT.PASSTHRUAUTH에 있는 모든 필드는 널(NULL) 입력이 불가능합니다.

표 77. SYSCAT.PASSTHRUAUTH 카탈로그 뷰에 있는 컬럼

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
GRANTOR	VARCHAR(128)		특권을 권한 부여한 사용자의 권한 부여 ID.
GRANTEE	VARCHAR(128)		특권을 보유한 그룹 또는 사용자 권한 부여 ID.
GRANTEETYPE	CHAR(1)		문자는 권한 받은 사용자 유형으로 지정합니다. U = 권한 받은 사용자가 개별 사용자임. G = 권한 받은 사용자가 그룹임.
SERVERNAME	VARCHAR(128)		사용자나 그룹에 권한이 부여되고 있는 데이터 소스의 이름.

SYSCAT.PROCEDURES

작성된 각 저장 프로시듀어에 대한 행을 포함합니다.

표 78. SYSCAT.PROCEDURES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
PROCSCHEMA	VARCHAR(128)		규정화된 프로시듀어 이름.
PROCNAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(18)		프로시듀어 인스턴스 이름(시스템을 생성할 수도 있습니다).
PROCEDURE_ID	INTEGER		저장 프로시듀어의 내부 ID.
DEFINER	VARCHAR(128)		프로시듀어 정의자의 권한 부여.
PARM_COUNT	SMALLINT		프로시듀어 매개변수의 수.
PARM_SIGNATURE	VARCHAR(180) FOR BIT DATA		내부 양식에서 매개변수 최대 90까지의 병합. 프로시듀어가 매개변수를 취할 경우 길이 0.
ORIGIN	CHAR(1)		사용자가 정의하면 항상 'E'
CREATE_TIME	TIMESTAMP		프로시듀어 등록의 시각.
DETERMINISTIC	CHAR(1)		Y = 결과가 결정됨 N = 결과가 결정되지 않음
FENCED	CHAR(1)		Y = 분리됨 N = 분리되지 않음
NULLCALL	CHAR(1)		항상 Y = NULLCALL
LANGUAGE	CHAR(8)		프로시듀어 내용의 구현 언어. 가능한 값은 다음과 같습니다. C COBOL JAVA SQL
IMPLEMENTATION	VARCHAR(254)	예	프로시듀어를 구현하는 경로/모듈/함수(LANGUAGE = C 또는 COBOL) 또는 메소드(LANGUAGE = JAVA)를 식별합니다.
CLASS	VARCHAR(128)	예	LANGUAGE = JAVA인 경우 이 프로시듀어를 구현하는 클래스를 식별합니다. 그렇지 않으면 널(NULL)입니다.

표 78. SYSCAT.PROCEDURES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
JAR_ID	VARCHAR(128)	예	If LANGUAGE = JAVA인 경우 이 프로시듀어를 구현하는 jar 파일을 식별합니다. 그렇지 않으면 널(NULL)입니다.
PARM_STYLE	CHAR(8)		DB2DARI = 언어는 C DB2GENRL = 언어는 Java DB2SQL = 언어는 C 또는 COBOL JAVA = 언어는 Java 또는 SQL GENERAL = 언어는 C 또는 COBOL GNLRNULL = 언어는 C 또는 COBOL
CONTAINS_SQL	CHAR(1)		프로시듀어가 SQL을 포함하는지 여부를 나타냅니다. C = CONTAINS SQL: SQL 데이터를 읽거나 수정하지 않는 SQL만 허용됩니다. M = MODIFY SQL DATA: 프로시듀에 허용된 모든 SQL이 허용됨 N = NO SQL: SQL이 허용되지 않음 R = READS SQL DATA: SQL 데이터를 읽는 SQL만 허용됩니다.
DBINFO	CHAR(1)		DBINFO 매개변수가 프로시듀어로 전달되었는지 여부를 나타냅니다. N = DBINFO를 전달하지 않음 Y = DBINFO를 전달함
PROGRAM_TYPE	CHAR(1)		프로시듀어가 어떤 식으로 호출되는지를 나타냅니다. M = 주 S = 서브루틴
RESULT_SETS	SMALLINT		리턴된 결과 집합의 평가 상위 한계.

표 78. SYSCAT.PROCEDURES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
VALID	CHAR(1)		공백 = SQL 프로시듀어가 아님 Y = SQL 프로시듀어가 유효함 N = SQL 프로시듀어가 유효하지 않음 X = 필요한 일부 함수 인스턴스가 삭제되었기 때문에 SQL 프로시듀어는 작동 불능 상태입니다. SQL 프로시듀어를 명백하게 삭제 및 재작성해야 합니다.
TEXT_BODY_OFFSET	INTEGER		SQL 프로시듀어인 경우, 이 컬럼에는 CREATE PROCEDURE문의 텍스트 전체에서 SQL 프로시듀어의 시작부분에 대한 오프셋이 포함됩니다. 외부 프로시듀어인 경우, 값은 0입니다.
TEXT	CLOB(1M)	예	SQL 프로시듀어인 경우, 이 컬럼에는 입력한 대로 정확하게 CREATE PROCEDURE문의 텍스트 전체가 포함됩니다. 텍스트 전체가 1M보다 길거나 외부 프로시듀어인 경우에는 널(NULL)입니다.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.PROCOPTIONS

각 행은 프로시듀어 특유의 옵션 값을 포함합니다.

표 79. SYSCAT.PROCOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
PROCSCHEMA	VARCHAR(128)		저장 프로시듀어 이름이나 별명에 대한 규정자.
PROCNAME	VARCHAR(128)		저장 프로시듀어의 이름이나 별명.
OPTION	VARCHAR(128)		저장 프로시듀어 옵션의 이름.
SETTING	VARCHAR(255)		저장 프로시듀어 옵션의 값.

SYSCAT.PROCPARMOPTIONS

각 행은 프로시듀어 매개변수에 고유한 옵션 값들을 포함합니다.

표 80. SYSCAT.PROCPARMOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
PROCSCHEMA	VARCHAR(128)		규정화된 프로시듀어 이름이나 별명.
PROCNAME	VARCHAR(128)		
ORDINAL	SMALLINT		프로시듀어 시그니쳐 내의 매개변수의 숫자 위치.
OPTION	VARCHAR(128)		저장 프로시듀어 옵션의 이름.
SETTING	VARCHAR(255)		값.

SYSCAT.PROCPARMS

저장 프로시듀어의 각 매개변수에 대한 행을 포함합니다.

표 81. SYSCAT.PROCPARMS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
PROCSCHEMA	VARCHAR(128)		규정화된 프로시듀어 이름.
PROCNAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(18)		프로시듀어 인스턴스 이름(시스템을 생성할 수도 있습니다).
SERVERNAME	VARCHAR(128)	예	저장 프로시듀어가 상주하는 데이터 소스의 이름.
ORDINAL	SMALLINT		프로시듀어 시그니쳐 내의 매개변수의 숫자 위치.
PARMNAME	VARCHAR(18)		매개변수 이름.
TYPESCHEMA	VARCHAR(128)		규정화된 매개변수의 데이터 유형.
TYPENAME	VARCHAR(18)		
TYPEID	SMALLINT	예	내부 유형 ID.
SOURCETYPEID	SMALLINT	예	소스 유형의 내부 유형 ID. 내장 유형의 경우에는 널(NULL).
NULLS	CHAR(1)		연합 데이터베이스 널(NULL) 입력 가능 규칙: Y = 널(NULL) 입력 가능 N = 널(NULL) 입력 불가능
LENGTH	INTEGER		매개변수의 길이.
SCALE	SMALLINT		매개변수의 스케일.
PARM_MODE	VARCHAR(5)		IN = 입력 OUT = 출력 INOUT = 입출력
CODEPAGE	SMALLINT		매개변수의 코드 페이지. 0은 FOR BIT DATA 속성으로 선언된 문자 데이터에 대한 매개변수이거나 적절하지 않음을 나타냅니다.
DBCS_CODEPAGE	SMALLINT	예	DBCS 코드 페이지. 숫자 필드의 경우에는 널(NULL).
AS_LOCATOR	CHAR(1)		항상 'N'

표 81. SYSCAT.PROCPARMS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
TARGET_TYPESCHEMA	VARCHAR(128)	예	매개변수의 유형이 참조이면 목표 행유형의 규정화된 이름을 포함합니다. 그렇지 않으면 널(NULL)입니다.
TARGET_TYPENAME	VARCHAR(18)		
SCOPE_TABSCHEMA	VARCHAR(128)	예	매개변수의 유형이 참조이면 범주(목표 테이블)의 규정화된 이름을 포함합니다. 그렇지 않으면 널(NULL)입니다.
SCOPE_TABNAME	VARCHAR(128)		

SYSCAT.REFERENCES

정의된 각 참조 제한조건에 대한 행을 포함합니다.

표 82. SYSCAT.REFERENCES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
CONSTNAME	VARCHAR(18)		제한조건 이름.
TABSCHEMA	VARCHAR(128)		규정화된 제한조건 이름.
TABNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		제한조건을 작성한 사용자.
REFKEYNAME	VARCHAR(18)		상위 키의 이름.
REFTABSCHEMA	VARCHAR(128)		상위 테이블의 이름.
REFTABNAME	VARCHAR(128)		
COLCOUNT	SMALLINT		외부 키의 컬럼 번호.
DELETERULE	CHAR(1)		삭제 규칙: A = NO ACTION C = CASCADE N = SET NULL R = RESTRICT
UPDATERULE	CHAR(1)		갱신 규칙: A = NO ACTION R = RESTRICT
CREATE_TIME	TIMESTAMP		참조 제한조건이 정의되는 때 시각.
FK_COLNAMES	VARCHAR (640)		외부 키 컬럼 이름의 목록. 경고: 이 컬럼은 나중에 제거될 것입니다. 이 정보에 대해서는 1329 페이지의 『SYSCAT.KEYCOLUSE』를 사용하십시오.
PK_COLNAMES	VARCHAR (640)		상위 키 컬럼 이름의 목록. 경고: 이 컬럼은 나중에 제거될 것입니다. 이 정보에 대해서는 1329 페이지의 『SYSCAT.KEYCOLUSE』를 사용하십시오.

주:

1. 버전 1에서 SYSCAT.REFERENCES 뷰는 SYSIBM.SYSRELS 테이블에 기초를 둡니다.

SYSCAT.REVTYPEMAPPINGS

각 행은 역 데이터 유형 맵핑(데이터 소스 데이터 유형에 지역으로 정의된 데이터 유형으로부터의 맵핑)을 포함합니다. 이 버전에는 데이터가 없습니다. 나중에 데이터 유형 맵핑과의 사용이 가능하도록 정의됩니다.

표 83. SYSCAT.REVTYPEMAPPINGS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
TYPE_MAPPING	VARCHAR(18)		역 유형 맵핑의 이름(시스템에 의해 생성될 수도 있습니다).
TYPESCHEMA	VARCHAR(128)	예	유형의 스키마 이름. 시스템 내장 유형의 경우에는 널(NULL).
TYPENAME	VARCHAR(18)		역 유형 맵핑에서 지역 유형의 이름.
TYPEID	SMALLINT		유형 식별자.
SOURCETYPEID	SMALLINT		소스 유형 식별자.
DEFINER	VARCHAR(128)		유형 맵핑이 작성될 때 권한 부여 ID.
LOWER_LEN	INTEGER	예	지역 유형의 길이/정밀도의 하위 바운드.
UPPER_LEN	INTEGER	예	지역 유형의 길이/정밀도의 상위 바운드. 널(NULL)이면 시스템이 최상의 길이/정밀도 속성을 결정합니다.
LOWER_SCALE	SMALLINT	예	지역 십진 데이터 유형을 위한 스케일의 하위 바운드.
UPPER_SCALE	SMALLINT	예	지역 십진 데이터 유형을 위한 스케일의 상위 바운드. 널(NULL)이면 시스템이 최상의 스케일 속성을 결정합니다.
S_OPR_P	CHAR(2)	예	지역 스케일과 지역 정밀도간의 관계. 기본적인 비교 연산자가 사용될 수 있습니다. 널(NULL)은 특별한 관계가 필요하지 않음을 나타냅니다.
BIT_DATA	CHAR(1)	예	Y = 유형이 2진 데이터용임. N = 유형이 2진 데이터용이 아님. NULL = 문자 데이터 유형이 아니거나 시스템을 비트 데이터 속성으로 결정함.
WRAPNAME	VARCHAR(128)	예	맵핑이 이 데이터 액세스 프로토콜에 적용합니다.
SERVERNAME	VARCHAR(128)	예	데이터 소스의 이름.
SERVERTYPE	VARCHAR (30)	예	맵핑이 이 유형의 데이터 소스에 적용합니다.

표 83. SYSCAT.REVTYPEMAPPINGS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
SERVERVERSION	VARCHAR(18)	예	맵핑이 이 버전의 SERVERTYPE에 적용합니다.
REMOTE_TYPESCHEMA	VARCHAR(128)	예	원격 유형의 스키마 이름
REMOTE_TYPENAME	VARCHAR(128)		데이터 소스에서 정의된 대로의 데이터 유형 이름.
REMOTE_META_TYPE	CHAR(1)	예	S = 원격 유형이 시스템 내장 유형임. T = 원격 유형이 구별 유형임.
REMOTE_LENGTH	INTEGER	예	원격 십진 유형에 대한 최대 자리 수 및 원격 문자 유형에 대한 최대 문자 수. 그렇지 않으면 널 (NULL).
REMOTE_SCALE	SMALLINT	예	십진 분리점의 오른쪽에 허용된 최대 자리 수(원격 십진 유형의 경우). 그렇지 않으면 널(NULL).
REMOTE_BIT_DATA	CHAR(1)	예	Y = 유형이 2진 데이터용임. N = 유형이 2진 데이터용이 아님. NULL = 문자 데이터 유형이 아니거나 시스템을 비트 데이터 속성으로 결정함.
USER_DEFINED	CHAR(1)		사용자에 의해 정의됨.
CREATE_TIME	TIMESTAMP		이 맵핑이 작성된 시간.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.SCHEMAAUTH

데이터베이스에서 특별한 스키마에 대한 특권을 권한 부여받은 그룹이나 사용자에 대해 하나 이상의 행을 포함합니다. 단일 행에 나타난 특정 권한 준 사용자가 특정 권한 받은 사용자에게 권한 부여한 단일 스키마 또는 모든 스키마 특권.

표 84. SYSCAT.SCHEMAAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
GRANTOR	VARCHAR(128)		SYSIBM 또는 특권을 권한 부여한 사용자의 권한 부여 ID.
GRANTEE	VARCHAR(128)		특권을 권한 부여한 그룹 또는 사용자의 권한 부여 ID.
GRANTEETYPE	CHAR(1)		U = 권한 받은 사용자가 개별 사용자임 G = 권한 받은 사용자가 그룹임
SCHEMANAME	VARCHAR(128)		스키마 이름.
ALTERINAUTH	CHAR(1)		다음은 권한 받은 사용자가 스키마에 대한 ALTERIN 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 G = 특권을 보유 및 권한 부여할 수 있음 N = 특권을 보유하지 않음
CREATEINAUTH	CHAR(1)		다음은 권한 받은 사용자가 스키마에 대한 CREATEIN 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 G = 특권을 보유 및 권한 부여할 수 있음 N = 특권을 보유하지 않음
DROPINAUTH	CHAR(1)		다음은 권한 받은 사용자가 스키마에 대한 DROPIN 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 G = 특권을 보유 및 권한 부여할 수 있음 N = 특권을 보유하지 않음

SYSCAT.SCHEMATA

각 스키마에 대한 행을 포함합니다.

표 85. SYSCAT.SCHEMATA 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
SCHEMANAME	VARCHAR(128)		스키마 이름.
OWNER	VARCHAR(128)		스키마의 권한 부여 ID. 내재적으로 작성된 스키마에 대한 값이 SYSIBM입니다.
DEFINER	VARCHAR(128)		스키마를 작성한 사용자.
CREATE_TIME	TIMESTAMP		오브젝트가 작성될 때를 나타내는 시각.
REMARKS	VARCHAR(254)	예	사용자 제공 주석.

SYSCAT.SERVEROPTIONS

각각의 행은 서버 레벨에 있는 구성 옵션을 포함합니다.

표 86. SYSCAT.SERVEROPTIONS 카탈로그 뷰에 있는 컬럼

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
WRAPNAME	VARCHAR(128)	예	랩퍼 이름.
SERVERNAME	VARCHAR(128)	예	서버의 이름.
SERVERTYPE	VARCHAR (30)	예	서버 유형
SERVERVERSION	VARCHAR(18)	예	서버 버전.
CREATE_TIME	TIMESTAMP		항목이 작성된 시간.
OPTION	VARCHAR(128)		서버 옵션의 이름.
SETTING	VARCHAR(2048)		서버 옵션의 값.
SERVEROPTIONKEY	VARCHAR(18)		행을 고유하게 식별.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.SERVERS

각각의 행은 데이터 소스를 나타냅니다. 이 카탈로그 테이블을 포함하는 동일한 인스턴스에 보관되는 테이블에는 카탈로그 항목이 필요하지 않습니다.

표 87. SYSCAT.SERVERS 카탈로그 뷰에 있는 컬럼들

이름	데이터 유형	널(NULL) 입력 가능	설명
WRAPNAME	VARCHAR(128)		랩퍼 이름.
SERVERNAME	VARCHAR(128)		시스템에 알려진 대로의 데이터 소스 이름.
SERVERTYPE	VARCHAR (30)	예	데이터 소스의 유형(항상 대문자).
SERVERVERSION	VARCHAR(18)	예	데이터 소스의 버전.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.STATEMENTS

데이터베이스에서 각 패키지의 각 SQL문에 대한 하나 이상의 행을 포함합니다.

표 88. SYSCAT.STATEMENTS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
PKGSCHEMA	VARCHAR(128)		패키지 이름.
PKGNAME	CHAR(8)		
STMTNO	INTEGER		응용프로그램의 소스 모듈에서 SQL문의 라인 번호.
SECTNO	SMALLINT		SQL문을 포함하는 패키지 섹션 수.
SEQNO	SMALLINT		항상 1입니다.
TEXT	CLOB(64K)		SQL문의 텍스트.

SYSCAT.TABAUTH

데이터베이스에서 특별한 테이블이나 뷰에 특권을 권한 부여한 각 사용자나 그룹에 대해 하나 이상의 행을 포함합니다. 단일 행에 나타난 특정 권한 준 사용자가 특정 권한 받은 사용자에게 권한 부여한 단일 테이블 또는 뷰의 특권.

표 89. SYSCAT.TABAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
GRANTOR	VARCHAR(128)		SYSIBM 또는 특권을 권한 부여한 사용자의 권한 부여 ID.
GRANTEE	VARCHAR(128)		특권을 부여한 그룹 또는 사용자의 권한 부여 ID.
GRANTEETYPE	CHAR(1)		U = 권한 받은 사용자가 개별 사용자임. G = 권한 받은 사용자가 그룹임.
TABSCHEMA	VARCHAR(128)		규정화된 테이블 또는 뷰 이름
TABNAME	VARCHAR(128)		
CONTROLAUTH	CHAR(1)		다음은 권한 받은 사용자가 테이블 또는 뷰에 대한 CONTROL 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음
ALTERAUTH	CHAR(1)		다음은 권한 받은 사용자가 테이블에 대한 ALTER 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음 G = 특권을 보유 및 권한 부여할 수 있음
DELETEAUTH	CHAR(1)		다음은 권한 받은 사용자가 테이블 또는 뷰에 대한 DELETE 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음 G = 특권을 보유 및 권한 부여할 수 있음

표 89. SYSCAT.TABAUTH 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
INDEXAUTH	CHAR(1)		다음은 권한 받은 사용자가 테이블에 대한 INDEX 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음 G = 특권을 보유 및 권한 부여할 수 있음
INSERTAUTH	CHAR(1)		다음은 권한 받은 사용자가 테이블 또는 뷰에 대한 INSERT 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음 G = 특권을 보유 및 권한 부여할 수 있음
SELECTAUTH	CHAR(1)		다음은 권한 받은 사용자가 테이블 또는 뷰에 대한 SELECT 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음 G = 특권을 보유 및 권한 부여할 수 있음
REFAUTH	CHAR(1)		다음은 권한 받은 사용자가 테이블 또는 뷰에 대한 REFERENCE 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음 G = 특권을 보유 및 권한 부여할 수 있음
UPDATEAUTH	CHAR(1)		다음은 권한 받은 사용자가 테이블 또는 뷰에 대한 UPDATE 특권을 보유하는지 나타냅니다. Y = 특권을 보유함 N = 특권을 보유하지 않음 G = 특권을 보유 및 권한 부여할 수 있음

SYSCAT.TABCONST

각 행은 테이블 제한조건 유형 점검, 고유, 기본 키 또는 외부 키를 나타냅니다.

표 90. SYSCAT.TABCONST 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
CONSTNAME	VARCHAR(18)		점검 제한조건 이름. (테이블 내의 고유성)
TABSCHEMA	VARCHAR(128)		제한조건이 적용되는 규정화된 테이블.
TABNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		정의된 제한조건에 따른 권한 부여 ID.
TYPE	CHAR(1)		제한조건 유형 지정: F = FOREIGN KEY K = CHECK P = PRIMARY KEY U = UNIQUE
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.TABLES

작성된 각 테이블, 뷰, 별명(nickname) 또는 별명(alias)에 대한 한 행을 포함합니다. 모든 카탈로그 테이블과 뷰는 SYSCAT.TABLES 카탈로그 뷰의 목록을 가집니다.

표 91. SYSCAT.TABLES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TABSCHEMA	VARCHAR(128)		테이블, 뷰, 별명(nickname) 또는 별명(alias)의 규정 이름.
TABNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		테이블, 뷰, 별명 또는 alias를 작성한 사용자.
TYPE	CHAR(1)		오브젝트 유형: A = 별명 H = 계층 테이블 N = 별명 S = 요약 테이블 T = 테이블 U = 유형화 테이블 V = 뷰 W = 유형화 뷰
STATUS	CHAR(1)		오브젝트 유형: N = 보통 테이블, 뷰, 별명 또는 별명 C = 테이블 또는 별명 점검 보류 X = 작동 불능 뷰 또는 별명
BASE_TABSCHEMA	VARCHAR(128)	예	TYPE = A인 경우, 이러한 컬럼은 이 별명에 의해 참조되는 테이블, 뷰, 별명(alias) 또는 별명(nickname)을 식별합니다. 그렇지 않으면 널(NULL)입니다.
BASE_TABNAME	VARCHAR(128)	예	
ROWTYPESCHEMA	VARCHAR(128)	예	이 테이블의 행 유형의 규정화된 이름을 포함합니다. 그렇지 않으면 널(NULL)입니다.
ROWTYPENAME	VARCHAR(18)		
CREATE_TIME	TIMESTAMP		오브젝트가 작성될 때를 나타내는 시각.
STATS_TIME	TIMESTAMP	예	이 테이블에 대한 통계를 기록될 때 변경된 마지막 시간. 통계가 사용 가능하지 않으면 널(NULL).
COLCOUNT	SMALLINT		테이블 내의 컬럼 수.

표 91. SYSCAT.TABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TABLEID	SMALLINT		내부 테이블 식별자.
TBSPACEID	SMALLINT		이 테이블에 대한 1차 테이블 공간의 내부 식별자.
CARD	BIGINT		테이블에 있는 행의 총 갯수. 테이블 계층의 테이블인 경우, 지정된 레벨의 계층에 있는 행의 수. 통계를 내지 않았거나 해당 행이 뷰나 별명을 설명하는 경우에는 -1. 계층 테이블(H 테이블)의 경우에는 -2.
NPAGES	INTEGER		테이블 행이 있는 총 페이지 수. 통계를 내지 않았거나 해당 행이 뷰나 별명을 설명하는 경우에는 -1. 서브테이블이나 H 테이블의 경우에는 -2.
FPAGES	INTEGER		총 페이지 수. 통계를 내지 않았거나 해당 행이 뷰나 별명을 설명하는 경우에는 -1. 서브테이블이나 H 테이블의 경우에는 -2.
OVERFLOW	INTEGER		테이블 행이 있는 총 오버플로우 레코드 수. 통계를 내지 않았거나 해당 행이 뷰나 별명을 설명하는 경우에는 -1. 서브테이블이나 H 테이블의 경우에는 -2.
TBSPACE	VARCHAR(18)	예	테이블에 대한 1차 테이블 공간 이름. 다른 테이블 공간이 지정되지 않으면, 테이블의 모든 부분은 이 테이블 공간에 저장됩니다. 별명들 및 뷰에 대해 널(NULL).
INDEX_TBSPACE	VARCHAR(18)	예	테이블에 대한 작성된 색인을 모두 가지고 있는 테이블 공간 이름. 별명과 뷰에 대해 또는 CREAATE TABLE문의 INDEX IN절과 동일한 값으로 지정되었거나 생략된 경우 널(NULL).
LONG_TBSPACE	VARCHAR(18)	예	테이블에 대한 긴 데이터(LONG 또는 LOB 컬럼 유형)를 모두 가지고 있는 테이블 공간 이름. 별명과 뷰 또는 CREATE TABLE문의 IN절과 동일한 값으로 지정되었거나 생략된 경우 널(NULL).
PARENTS	SMALLINT	예	테이블의 상위 테이블 수(이 테이블이 종속되는 참조 제한조건의 수).
CHILDREN	SMALLINT	예	테이블의 종속 테이블 수(이 테이블이 상위 테이블이 되는 참조 제한조건의 수).
SELFREFS	SMALLINT	예	테이블에 대한 자기 참조 제한조건(상위 테이블 및 종속 테이블 둘다 있는 참조 제한조건의 수).
KEYCOLUMNS	SMALLINT	예	테이블 기본 키 내의 컬럼 수.
KEYINDEXID	SMALLINT	예	1차 색인의 색인 ID. 기본 키가 없는 경우 널(NULL) 또는 0.

표 91. SYSCAT.TABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
KEYUNIQUE	SMALLINT		테이블에 정의된 고유 제한조건(기본 키 제외)의 수.
CHECKCOUNT	SMALLINT		테이블에 정의된 점검 제한조건.
DATACAPTURE	CHAR(1)		Y = 테이블이 데이터 복제에 관여함 N = 관여하지 않음 L = 테이블이 LONG VARCHAR 및 LONG VARGRAPHIC 컬럼 복제를 포함한 데이터 복제에 관여함
CONST_CHECKED	CHAR(32)		바이트 1은 외부 키 제한조건을 나타냅니다. 바이트 2는 점검 제한조건을 나타냅니다. 바이트 5는 요약 테이블을 나타냅니다. 바이트 6 생성된 컬럼을 나타냅니다. 다른 바이트들은 예약됩니다. 점검 중 제한조건 정보를 코드화합니다. 다음과 같은 값이 있습니다. Y = 시스템에 의한 점검 U = 사용자에 의한 점검 N = 점검하지 않음(보류) W = 테이블이 점검 보류(보류 중) 상태라면 'U' 상태
PMAP_ID	SMALLINT	예	테이블이 사용한 파티션 맵핑의 식별자. 별명들 및 뷰에 대해 널(NULL).
PARTITION_MODE	CHAR(1)		파티션된 데이터베이스에 있는 테이블에 사용되는 모드. H = 파티션 키에 해쉬됨 R = 테이블이 데이터베이스 파티션 전반에 복제됨 정의된 파티션 키가 없는 단일 파티션 노드 그룹에 있는 테이블, 뷰 및 별명인 경우 공백. 별명의 경우에도 공백.
LOG_ATTRIBUTE	CHAR(1)		0 = 기본 로그 1 = 작성된 테이블에 초기에 로그되지 않음
PCTFREE	SMALLINT		이후 삽입에 대해 예약될 각 페이지의 비율. ALTER TABLE를 변경할 수 있습니다.

표 91. SYSCAT.TABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
APPEND_MODE	CHAR(1)		페이지에 얼마나 많은 행이 삽입되는지를 제어합니다. N = 사용 가능한 경우 새 행이 기존 공간에 삽입됨 Y = 새 행이 데이터의 끝에 추가됨 초기 값은 N입니다.
REFRESH	CHAR(1)		새로 고침 모드 D = 연기 I = 즉시 실행 O = 한번만 요약 테이블이 아니라면 공백
REFRESH_TIME	TIMESTAMP	예	REFRESH = D 또는 O의 경우, 마지막으로 데이터를 갱신한 REFRESH TABLE문의 시간소인. 그렇지 않으면 널(NULL).
LOCKSIZE	CHAR(1)		DML문에 의해 액세스될 때 테이블에 대해 선호하는 잠금 입도를 나타냅니다. 테이블에만 적용합니다. 가능한 값은 다음과 같습니다. R = 행 T = 테이블 적용 가능하지 않은 경우에는 공백 초기 값은 R입니다.
VOLATILE	CHAR(1)		C = 테이블의 기본 행수는 VOLATILE 적용 가능하지 않은 경우에는 공백
REMARKS	VARCHAR(254)	예	사용자 제공 주석.

SYSCAT.TABLESPACES

각 테이블 공간에 대한 행을 포함합니다.

표 92. SYSCAT.TABLESPACES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TBSPACE	VARCHAR(18)		테이블 공간 이름.
DEFINER	VARCHAR(128)		테이블 공간 정의자의 권한 부여 ID
CREATE_TIME	TIMESTAMP		테이블 공간 작성 시간.
TBSPACEID	INTEGER		내부 테이블 공간 식별자.
TBSPACETYPE	CHAR(1)		테이블 공간의 유형: S = 시스템 관리 공간 D = 데이터베이스 관리 공간
DATATYPE	CHAR(1)		저장될 데이터의 유형: A = 모든 영구 데이터 유형 L = LONG 데이터만 T = SYSTEM TEMPORARY 테이블만 U = 선언된 임시 테이블만
EXTENTSIZE	INTEGER		PAGESIZE *크기*의 페이지 단위로 확장 *크기*. 다음 컨테이너로 전환하기 전에 많은 페이지가 테이블 공간의 한 컨테이너에 기록됩니다.
PREFETCHSIZE	INTEGER		프리페치 수행시 읽혀질 PAGESIZE *크기*의 페이지 수.
OVERHEAD	DOUBLE		제어기 오버헤드와 디스크 탐색 및 밀리세컨드로 대기 시간.
TRANSFERRATE	DOUBLE		PAGESIZE *크기*의 한 페이지를 버퍼에 일어들일 시간.
PAGESIZE	INTEGER		테이블 공간의 페이지 *크기*(바이트)
NGNAME	VARCHAR(18)		테이블 공간에 대한 노드 이름.
BUFFERPOOLID	INTEGER		테이블 공간이 사용한 버퍼 풀의 ID(1은 기본 버퍼 풀을 나타냅니다.
DROP_RECOVERY	CHAR(1)		N = 테이블이 DROP TABLE문 이후 복구 불가능함 Y = 테이블이 DROP TABLE문 이후 복구 가능함

표 92. SYSCAT.TABLESPACES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
REMARKS	VARCHAR(254)	예	사용자 제공 주석.

SYSCAT.TABOPTIONS

각 행은 원격 테이블과 연관된 옵션을 포함합니다.

표 93. SYSCAT.TABOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TABSCHEMA	VARCHAR(128)		테이블, 뷰, 별명 또는 별명의 규정화된 이름.
TABNAME	VARCHAR(128)		
OPTION	VARCHAR(128)		테이블, 뷰, 별명 또는 별명 옵션의 이름.
SETTING	VARCHAR(255)		값.

SYSCAT.TBSPACEAUTH

데치터베이스에서 특별한 테이블 공간에 USE 특권을 권한 부여받은 각 사용자 또는 그룹에 대해 하나의 행을 포함합니다.

표 94. SYSCAT.TBSPACEAUTH 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
GRANTOR	CHAR(128)		SYSIBM 또는 특권을 권한 부여한 사용자의 권한 부여 ID.
GRANTEE	CHAR(128)		특권을 부여한 그룹 또는 사용자의 권한 부여 ID.
GRANTEETYPE	CHAR(1)		U = 권한 받은 사용자가 개별 사용자임 G = 권한 받은 사용자가 그룹임
TBSPACE	VARCHAR(18)		테이블 공간의 이름.
USEAUTH	CHAR(1)		다음은 권한 받은 사용자가 테이블 공간에 대한 USE 특권을 보유하는지 나타냅니다. G = 특권을 보유 및 권한 부여할 수 있음 N = 특권을 보유하지 않음 Y = 특권을 보유함

SYSCAT.TRIGDEP

일부 다른 오브젝트에서 트리거의 모든 종속성에 대한 행을 포함합니다.

표 95. SYSCAT.TRIGDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TRIGSCHEMA	VARCHAR(128)		규정화된 트리거 이름.
TRIGNAME	VARCHAR(18)		
BTYPE	CHAR(1)		오브젝트 BNAME의 유형: A = 별명 F = 함수 인스턴스 N = 별명 O = 테이블이나 뷰 계층에서 모든 서브테이블 또는 서브뷰에 대한 특권 종속성 R = 구조화 유형 S = 요약 테이블 T = 테이블 U = 유형화 테이블 V = 뷰 W = 유형화 뷰 X = 확장 색인
BSCHEMA	VARCHAR(128)		트리거가 종속된 규정화된 오브젝트 이름.
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	예	BTYPE= O, S, T, U, V 또는 W인 경우, 이 트리거에 의해 요구된 테이블 또는 뷰의 특권을 코드화합니다. 그렇지 않으면, 널(NULL)입니다.

SYSCAT.TRIGGERS

각 트리거에 대한 행을 포함합니다. 테이블 계층의 경우, 각 트리거는 그 트리거가 작성된 계층의 레벨에서만 기록됩니다.

표 96. SYSCAT.TRIGGERS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
TRIGSCHEMA	VARCHAR(128)		규정화된 트리거 이름.
TRIGNAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		트리거가 정의된 상태에서 권한 부여.
TABSCHEMA	VARCHAR(128)		트리거가 적용되는 규정화된 테이블 이름.
TABNAME	VARCHAR(128)		
TRIGTIME	CHAR(1)		트리거 조치가 트리거를 자극한 이벤트와 관련되어 기본 테이블에 적용되는 시간 A = 이벤트 이후 트리거 적용 B = 이벤트 이전 트리거 적용
TRIGEVENT	CHAR(1)		트리거를 자극한 이벤트. I = 삽입 D = 삭제 U = 갱신
GRANULARITY	CHAR(1)		트리거는 다음과 같이 단 한번으로 실행됩니다. S = 명령문 R = 행
VALID	CHAR(1)		Y = 트리거가 유효함 X = 트리거가 작동 불능이므로 재작성해야 함
CREATE_TIME	TIMESTAMP		트리거가 정의되는 때의 시간. 함수와 유형을 분석하는 데 사용됩니다.
QUALIFIER	VARCHAR(128)		오브젝트 정의시의 기본 스키마의 값을 포함합니다.
FUNC_PATH	VARCHAR(254)		트리거가 정의되는 때의 함수 경로. 함수와 유형을 분석하는 데 사용됩니다.
TEXT	CLOB(64K)		입력된 것과 동일한, CREATE TRIGGER문의 전체 텍스트.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.TYPEMAPPINGS

각 행은 지역 내장 데이터 유형에 대한 원격 내장 데이터 유형의 사용자 정의 맵핑을 포함합니다.

표 97. SYSCAT.TYPEMAPPINGS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
TYPE_MAPPING	VARCHAR(18)		유형 맵핑의 이름(시스템에 의해 생성될 수도 있습니다).
TYPESCHEMA	VARCHAR(128)	예	유형의 스키마 이름. 시스템 내장 유형의 경우에는 널(NULL).
TYPENAME	VARCHAR(18)		데이터 유형 맵핑에서 지역 유형의 이름.
TYPEID	SMALLINT		유형 식별자.
SOURCETYPEID	SMALLINT		소스 유형 식별자.
DEFINER	VARCHAR(128)		유형 맵핑이 작성될 때 권한 부여 ID.
LENGTH	INTEGER	예	데이터 정밀도의 최대 길이. 널(NULL)이면 시스템이 최상의 길이/정밀도를 결정합니다.
SCALE	SMALLINT	예	DECIMAL 필드에 대한 스케일. 널(NULL)이면 시스템이 최상의 스케일 속성을 결정합니다.
BIT_DATA	CHAR(1)	예	Y = 유형이 2진 데이터용임. N = 유형이 2진 데이터용이 아님. NULL = 문자 데이터 유형이 아니거나 시스템을 비트 데이터 속성으로 결정함.
WRAPNAME	VARCHAR(128)	예	맵핑이 이 데이터 액세스 프로토콜에 적용합니다.
SERVERNAME	VARCHAR(128)	예	데이터 소스의 이름.
SERVERTYPE	VARCHAR (30)	예	맵핑이 이 유형의 데이터 소스에 적용합니다.
SERVERVERSION	VARCHAR(18)	예	맵핑이 이 버전의 SERVERTYPE에 적용합니다.
REMOTE_TYPESCHEMA	VARCHAR(128)	예	원격 유형의 스키마 이름
REMOTE_TYPENAME	VARCHAR(128)		데이터 소스에서 정의된 대로의 데이터 유형 이름.
REMOTE_META_TYPE	CHAR(1)	예	S = 원격 유형이 시스템 내장 유형임. T = 원격 유형이 구별 유형임.

표 97. SYSCAT.TYPEMAPPINGS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
REMOTE_LOWER_LEN	INTEGER	예	원격 십진 유형의 길이/정밀도의 하위 바운드. 문자 데이터 유형의 경우, 이 필드는 문자 수를 나타냅니다.
REMOTE_UPPER_LEN	INTEGER	예	원격 십진 유형의 길이/정밀도의 상위 바운드. 문자 데이터 유형의 경우, 이 필드는 문자 수를 나타냅니다.
REMOTE_LOWER_SCALE	SMALLINT	예	원격 유형의 스케일의 하위 바운드.
REMOTE_UPPER_SCALE	SMALLINT	예	원격 유형의 스케일의 상위 바운드.
REMOTE_S_OPR_P	CHAR(2)	예	원격 스케일과 원격 정밀도간의 관계. 기본적인 비교 연산자가 사용될 수 있습니다. 특정 관계가 필요하지 않음을 나타내는 널(NULL)은 필요하지 않습니다.
REMOTE_BIT_DATA	CHAR(1)	예	Y = 유형이 2진 데이터용임. N = 유형이 2진 데이터용이 아님. NULL = 문자 데이터 유형이 아니거나 시스템을 비트 데이터 속성으로 결정함.
USER_DEFINED	CHAR(1)		사용자에 의해 제공된 정의.
CREATE_TIME	TIMESTAMP		이 맵핑이 작성된 시간.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSCAT.USEROPTIONS

각 행은 서버 특유의 옵션 값을 포함합니다.

표 98. SYSCAT.USEROPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
AUTHID	VARCHAR(128)		지역 권한 부여 ID(항상 대문자)
SERVERNAME	VARCHAR(128)		사용자가 정의된 해당 서버의 이름.
OPTION	VARCHAR(128)		사용자 옵션의 이름.
SETTING	VARCHAR(255)		값.

SYSCAT.VIEWDEP

일부 다른 오브젝트 뷰 또는 요약 테이블의 모든 종속성에 대한 행을 포함합니다. 또한 이 뷰에 대한 특권이 기반 테이블과 뷰에 대한 특권에 종속되는 방법을 코드화합니다.

표 99. SYSCAT.VIEWDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
VIEWSCHEMA	VARCHAR(128)		기본 테이블에 종속되는 뷰의 이름 또는 요약 테이블의 이름.
VIEWNAME	VARCHAR(128)		
DTYPE	CHAR(1)		S = 요약 테이블 V = 뷰(유형화되지 않음) W = 유형화 뷰
DEFINER	VARCHAR(128)	예	뷰 작성자의 권한 부여 ID.
BTYPE	CHAR(1)		오브젝트 BNAME의 유형: A = 별명 F = 함수 인스턴스 N = 별명 O = 테이블이나 뷰 계층에서 모든 서브테이블 또는 서브뷰에 대한 특권 종속성 I = 기본 테이블에 대한 종속성을 기록하는 경우 색인 R = 구조화 유형 S = 요약 테이블 T = 테이블 U = 유형화 테이블 V = 뷰 W = 유형화 뷰
BSCHEMA	VARCHAR(128)		뷰가 종속된 규정화된 오브젝트 이름.
BNAME	VARCHAR(128)		

표 99. SYSCAT.VIEWDEP 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
TABAUTH	SMALLINT	예	BTYPE = O, S, T, U, V, W이면, 이 뷰가 종속되는 기초가 되는 테이블이나 뷰에 대한 특권을 코드화합니다. 그렇지 않으면 널(NULL).

SYSCAT.VIEWS

작성된 각 뷰에 대한 하나 이상의 행을 포함합니다.

표 100. SYSCAT.VIEWS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
VIEWSCHEMA	VARCHAR(128)		요약 테이블을 정의하는 데 사용되는 뷰 또는 테이블의 이름.
VIEWNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		뷰 작성자의 권한 부여 ID.
SEQNO	SMALLINT		항상 1입니다.
VIEWCHECK	CHAR(1)		뷰 유형 상태 점검: N = 아니오 점검 옵션 L = 지역 점검 옵션 C = 연쇄 점검 옵션
READONLY	CHAR(1)		Y = 뷰는 그 정의 때문에 읽기 전용입니다. N = 뷰가 읽기 전용이 아닙니다.
VALID	CHAR(1)		Y = 뷰 또는 요약 테이블 정의가 유효함 X = 뷰 또는 요약 테이블 정의가 작동 불능이므로 재작성해야 함.
QUALIFIER	VARCHAR(128)		오브젝트 정의시의 기본 스키마의 값을 포함합니다.
FUNC_PATH	VARCHAR(254)		뷰를 정의할 시점의 뷰 작성자의 SQL 경로. 뷰가 데이터 조작문에 사용될 때, 이 경로는 뷰의 함수 호출을 분석하는 데 사용됩니다. 버전 2전에 작성된 뷰에 대해 SYSIBM.
TEXT	CLOB(64k)		CREATE VIEW문의 텍스트.

SYSCAT.WRAPOPTIONS

각 행은 랩퍼 특유의 옵션을 포함합니다.

표 101. SYSCAT.WRAPOPTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
WRAPNAME	VARCHAR(128)		랩퍼 이름.
OPTION	VARCHAR(128)		랩퍼 옵션의 이름.
SETTING	VARCHAR(255)		값.

SYSCAT.WRAPPERS

각 행은 등록된 랩퍼에 대한 정보를 포함합니다.

표 102. SYSCAT.WRAPPERS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명
WRAPNAME	VARCHAR(128)		랩퍼 이름.
WRAPTYPE	CHAR(1)		N = 비 관계형 R = 관계형
WRAPVERSION	INTEGER		랩퍼의 버전.
LIBRARY	VARCHAR(255)		이 랩퍼와 연관된 데이터 소스와 통신하기 위해 사용된 코드를 포함하는 파일의 이름.
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

SYSSTAT.COLDIST

각 행은 N번째로 가장 빈번한 값 또는 일부 컬럼의 정량 값. 유형화된 테이블의 상속된 컬럼에 대해서는 통계가 기록되지 않습니다.

표 103. SYSSTAT.COLDIST 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
TABSCHEMA	VARCHAR (128)		이 항목이 적용되는 규정화 된 테이블 이름.	
TABNAME	VARCHAR (128)			
COLNAME	VARCHAR (128)		이 항목이 적용되는 컬럼 이름.	
TYPE	CHAR(1)		수집된 통계 유형: F = 빈도(자주 사용되는 값) Q = Quantile 값	
SEQNO	SMALLINT		TYPE = F인 경우, 이 컬럼의 N은 N번째의 가장 자주 사용되는 값을 식별합니다. TYPE = Q인 경우, 이 컬럼의 N은 N번째의 Quantile 값을 식별합니다.	
COLVALUE	VARCHAR (254)	예	문자 리터럴 또는 널(NULL) 값으로서, 데이터 값. 이 컬럼은 통계와 관련된 컬럼에 적절한 값의 유효한 표현으로 갱신할 수 있습니다. 빈도 값에 널(NULL)이 필요한 경우, 컬럼은 NULL로 설정되야 합니다.	예
VALCOUNT	BIGINT		TYPE = F인 경우, VALCOUNT는 컬럼에서 COLVALUE의 발생 수입니다. TYPE = Q인 경우, VALCOUNT는 그 값이 COLVALUE보다 작거나 같은 행의 수입니다. 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • >= 0(제로)	예
DISTCOUNT	BIGINT		TYPE = q인 경우, 이 컬럼은 COLVALUE보다 작거나 같은 구별 값의 수를 기록합니다(사용할 수 없는 경우 널(NULL)). 값이 COLVALUE보다 작거나 같은 행의 수.	예

SYSSTAT.COLUMNS

통계를 갱신할 수 있는 각 컬럼마다 한 행을 포함합니다. 유형화된 테이블의 상속된 컬럼에 대해서는 통계가 기록되지 않습니다.

표 104. SYSSTAT.COLUMNS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
TABSCHEMA	VARCHAR (128)		컬럼을 포함하는 규정화된 테이블 이름.	
TABNAME	VARCHAR (128)			
COLNAME	VARCHAR (128)		컬럼 이름	
COLCARD	BIGINT		컬럼의 구별 값 수. 통계를 내지 않은 경우 -1. H 테이블의 컬럼과 계승된 컬럼의 경우 -2. 임의의 컬럼에 대해, COLCARD는 그 컬럼을 포함하는 테이블의 기본 행수보다 더 높은 값을 가질 수 없습니다. 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
HIGH2KEY	VARCHAR (33)	예	두 번째로 높은 컬럼 값. 통계가 충계되지 않은 경우와 H 테이블의 상속된 컬럼의 경우에는 이 필드가 공백입니다. 이 컬럼은 통계와 관련된 컬럼에 적절한 값의 유효한 표현으로 갱신할 수 있습니다. LOWKEY2는 HIGH2KEY보다 클 수 없습니다.	예

표 104. SYSSTAT.COLUMNS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
LOW2KEY	VARCHAR (33)	예	두 번째로 낮은 컬럼 값. 통계가 충계되지 않은 경우와 H 테이블의 상속된 컬럼의 경우에는 공백입니다. 이 컬럼은 통계와 관련된 컬럼에 적절한 값의 유효한 표현으로 갱신할 수 있습니다.	예
AVGCOLLEN	INTEGER		평균 컬럼 길이. LONG 필드 또는 LOB 나 통계를 내지 않은 경우 -1. H 테이블의 컬럼과 계승된 컬럼의 경우 -2. 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
NUMNULLS	BIGINT		컬럼에 있는 널(NULL)의 갯수가 들어 있습니다. 통계를 내지 않았을 경우에는 -1. 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예

SYSSTAT.FUNCTIONS

각 사용자 정의 함수(스칼라 또는 총계)에 대한 행을 포함합니다. 내장 함수를 포함하지 않습니다. 유형화된 테이블의 상속된 컬럼에 대해서는 통계가 기록되지 않습니다.

표 105. SYSSTAT.FUNCTIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
FUNCSCHEMA	VARCHAR (128)		규정화된 함수 이름.	
FUNCNAME	VARCHAR (18)			
SPECIFICNAME	VARCHAR (18)		함수 특정(인스턴스) 이름.	
IOS_PER_INVOC	DOUBLE		호출당 입출력의 평가 수. 알려져 있지 않을 경우에는 -1(기본값은 0). 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
INSTS_PER_INVOC	DOUBLE		호출당 명령의 평가 수; 알지 못하면 -1 (450 기본값). 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
IOS_PER_ARGBYTE	DOUBLE		입력 인수 바이트당 입출력의 평가 수; 알지 못하면 -1(0 기본값). 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
INSTS_PER_ARGBYTE	DOUBLE		입력 인수 바이트당 명령의 평가 수; 알지 못하면 -1(0 기본값). 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예

표 105. SYSSTAT.FUNCTIONS 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
PERCENT_ARGBYTES	SMALLINT		함수가 실제로 읽는 입력 인수 바이트의 평가 평균 퍼센트. 알지 못하면 -1(100 기본값). 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 100 - 0(제로) 사이	예
INITIAL_IOS	DOUBLE		처음/마지막으로 호출된 함수를 수행한 입출력의 평가 수. 알지 못하면 -1(0 기본값). 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
INITIAL_INSTS	DOUBLE		처음/마지막으로 호출된 함수를 실행한 명령의 평가 수. 알지못하면 -1 (0 기본값). 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
CARDINALITY	BIGINT		테이블 함수의 예측된 기본 행 수. 알려져 있지 않거나 함수가 테이블 함수가 아닌 경우에는 -1.	예
SELECTIVITY	DOUBLE		사용자 정의 술어에 사용됩니다. 사용자 정의 술어가 없을 경우, 기본값 = -1. 주1 참조.	

주:

1. 이 컬럼은 모든 사용자 정의 함수에 대한 시스템 카탈로그에서 DB2 버전 5.2로부터 6.1로 이중되는 동안 -1로 설정됩니다. 사용자 정의 술어의 경우, 시스템 카탈로그의 선택성은 -1이 됩니다. 이 경우, 최적화 알고리즘에서 사용하는 선택성 값은 0.01.입니다.

SYSSTAT.INDEXES

테이블에 대해 정의된 각 색인에 대해 한 행을 포함합니다.

표 106. SYSSTAT.INDEXES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
INDSCHEMA	VARCHAR (128)		규정화된 색인 이름.	
INDNAME	VARCHAR (18)			
TABSCHEMA	VARCHAR (128)		테이블 이름의 규정자.	
TABNAME	VARCHAR (128)		색인이 정의되는 테이블이나 별명의 이름.	
COLNAMES	CLOB(1M)		+ 또는 - 접두부가 있는 컬럼 이름 목록.	
NLEAF	INTEGER		리프 페이지의 수; 통계를 내지 않았을 경우에는 -1 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 > 0 (제로)	예
NLEVELS	SMALLINT		색인 레벨 수; 통계를 내지 않았을 경우에는 -1 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 > 0 (제로)	예
FIRSTKEYCARD	BIGINT		구별되는 첫번째 키 값의 수. 통계를 내지 않았을 경우에는 -1. 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예

표 106. SYSSTAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
FIRST2KEYCARD	BIGINT		색인의 처음 두 컬럼을 사용하는 구별 키의 수(통계를 내지 않았거나 적용 불가능한 경우에는 -1) 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
FIRST3KEYCARD	BIGINT		색인의 처음 세 컬럼을 사용하는 구별 키의 수(통계를 내지 않았거나 적용 불가능한 경우에는 -1) 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
FIRST4KEYCARD	BIGINT		색인의 처음 네 컬럼을 사용하는 구별 키의 수(통계를 내지 않았거나 적용 불가능한 경우에는 -1) 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
FULLKEYCARD	BIGINT		구별되는 전체 키 값의 수. 통계를 내지 않았을 경우에는 -1. 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
CLUSTERRATIO	SMALLINT		이것은 최적화 알고리즘에서 사용됩니다. 색인의 데이터 클러스터링 등급을 나타냅니다. 통계를 내지 않았거나 세부사항 색인 통계를 낸 경우에는 -1. 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 0 - 100 사이	예

표 106. SYSSTAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
CLUSTERFACTOR	DOUBLE		이것은 최적화 알고리즘에서 사용됩니다. 클러스터링 등급의 더 정교한 측정입니다. 세부사항 색인 통계를 내지 않은 경우에는 -1. 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 0 - 1 사이	예
SEQUENTIAL_PAGES	INTEGER		간격 사이에 큰 간격이 없거나 거의 없을 경우, 색인 키 순서의 디스크에 지정된 리프 페이지의 수(사용 가능한 통계가 없을 경우에는 -1). 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 >= 0 (제로)	예
DENSITY	INTEGER		색인이 차지한 페이지 범위에서 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로서, 퍼센트로 표시됩니다(0에서 100까지의 정수로 표시되고, 사용 가능한 통계가 없을 경우에는 -1임). 이 컬럼은 다음 값으로만 갱신될 수 있습니다. • -1 또는 0 - 100 사이	예

표 106. SYSSTAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
PAGE_FETCH_PAIRS	VARCHAR (254)		문자 양식에서 표현된, 쌍으로 된 정수의 목록. 각 쌍은 가설 버퍼에 페이지 수를 나타내며 페이지 페치의 수는 가설 버퍼를 사용하여 색인을 스캔하는 데 필요합니다(사용 가능한 데이터가 없을 경우 문자열 길이 0) 이 컬럼은 다음 값으로 갱신될 수 있습니다. • 쌍으로 된 분리 문자와 구분자 문자는 숫자가 아닌 문자입니다. • 공백은 쌍으로 된 분리 문자와 구분자로 간주된 문자만 가능합니다. • 각 숫자 항목은 쌍 구분자 문자에서 분리된 상대 숫자 항목을 동반해야 합니다. • 각 쌍은 쌍 분리 문자에 의해 다른 쌍과 분리되어야 합니다. • 예상된 각 숫자 항목은 0-9 사이여야 합니다(양수 값만).	예

SYSSTAT.TABLES

각 *기본* 테이블에 대한 한 행을 포함합니다. 그러므로 뷰 또는 별명은 포함되지 않습니다. 입력된 테이블의 경우, 테이블 계층 중 루트 테이블만이 이 뷰에 포함됩니다. 유형화된 테이블의 상속된 컬럼에 대해서는 통계가 기록되지 않습니다. CARD 값은 다른 통계가 테이블 계층 전체에 적용될 경우에만 루트 테이블에 적용됩니다.

표 107. SYSSTAT.TABLES 카탈로그 뷰

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
TABSCHEMA	VARCHAR (128)		규정화된 테이블 이름.	
TABNAME	VARCHAR (128)			
CARD	BIGINT		테이블에 있는 총 행 수. 통계를 내지 않았을 경우에는 -1. 테이블에 대해 CARD의 갱신은 테이블 컬럼의 임의 COLCARD 값보다 작은 값을 지정하려고 해서는 안 됩니다. 이 컬럼은 다음 값으로만 갱신할 수 있습니다. [115]. • -1 또는 >= 0 (제로)	예
NPAGES	INTEGER		테이블 행이 있는 총 페이지 수. 통계를 내지 않은 경우에는 -1. 서브테이블과 H 테이블의 경우에는 -2. 이 컬럼은 다음 값으로만 갱신할 수 있습니다. [115] • -1 또는 >= 0 (제로)	예
FPAGES	INTEGER		파일에 있는 총 페이지 수. 통계를 내지 않은 경우에는 -1. 서브테이블과 H 테이블의 경우에는 -2. 이 컬럼은 다음 값으로만 갱신할 수 있습니다. [115] • -1 또는 >= 0 (제로)	예

115. -2 값은 변경할 수 없으며 컬럼 값을 직접 -2로 설정할 수 없습니다.

표 107. SYSSTAT.TABLES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능	설명	갱신 가능
OVERFLOW	INTEGER		테이블에 있는 총 오버플로우 레코드 수. 통계를 내지 않은 경우에는 -1. 서브테이블과 H 테이블의 경우에는 -2. 이 컬럼은 다음 값으로만 갱신할 수 있습니다: [115] • -1 또는 >= 0 (제로)	예

부록E. 구조화 유형 사용을 위한 카탈로그 뷰

확장 색인을 사용하는 경우, 추가 카탈로그 뷰는 SYSCAT 카탈로그 뷰를 보완하는 유용한 정보를 제공합니다. 이러한 뷰는 자동으로 작성되지 않습니다. 이 뷰는 OBJCAT 스키마에 작성되고 기본값으로 모든 뷰에 대한 SELECT 특권이 공용에 부여됩니다.

경고: 이러한 뷰 집합은 카탈로그 이주를 지원하는 다음 버전까지만 임시로 사용됩니다. 응용프로그램에서는 이러한 뷰가 모든 데이터베이스에 있다고 가정해서는 안되며 이러한 카탈로그 뷰는 차후 버전에서 제공되지 않을 수도 있음을 고려해야 합니다. 이 뷰의 정보는 차후 버전에서 SYSCAT 뷰를 통해 지원됩니다.

보기는 다음 단계에 따라 작성할 수 있습니다.

- 명령행 처리기를 사용하여 SYSADM이나 DBADM 권한을 갖는 권한 부여 ID로 데이터베이스에 연결하십시오.
- 사용자가 DB2 인스턴스의 홈 디렉토리에 있는지 확인하십시오.
- UNIX 기반 시스템에서 다음 명령을 발행하십시오.

```
db2 -tvf sqllib/bin/objcat.db2
```

- OS/2 또는 Windows 기반 시스템에서 다음 명령을 발행하십시오.

```
db2 -tvf sqllib\bin\objcat.db2
```

objcat.db2으로 생성된 보기는 다음 단계에 따라 제거시킬 수 있습니다.

- 명령행 처리기를 사용하여 SYSADM이나 DBADM 권한을 갖는 권한 부여 ID로 데이터베이스에 연결하십시오.
- 사용자가 DB2 인스턴스의 홈 디렉토리에 있는지 확인하십시오.
- UNIX 기반 시스템에서 다음 명령을 발행하십시오.

```
db2 -tvf sqllib/bin/objcatdp.db2
```

- OS/2 또는 Windows 기반 시스템에서 다음 명령을 발행하십시오.

```
db2 -tvf sqllib\bin\objcatdp.db2
```

주: 데이터베이스에 이미 스키마 OBJCAT가 있는 경우, 사용자 소유의 **objcat.db2** 파일 사본을 작성하고 두 번째와 세 번째 CREATE SCHEMA 문의 스키마 이름을 적절한 이름으로 변경해야 합니다.

OBJCAT.DB2 파일에 있는 명령문은 모든 추가 OBJCAT 카탈로그 뷰를 생성합니다.

이 부록에는 각 OBJCAT 카탈로그 뷰의 설명이 포함되어 있습니다. 관련 SYSCAT 뷰에 대해서는 1281 페이지의 『부록D. 카탈로그 뷰』에서 참조하십시오.

카탈로그 뷰는 SQL 데이터 정의문, 환경 루틴 및 특정 유틸리티에 응답하는 정상 조작중에 갱신됩니다. 카탈로그 뷰의 데이터는 정상 SQL 조회 기능을 통하여 사용 가능합니다. 기술하는 오브젝트 유형에 따라 컬럼은 일관된 이름을 가집니다.

서술된 오브젝트	컬럼 이름
테이블	TABSCHEMA, TABNAME
색인	INDSCHEMA, INDNAME
뷰	VIEWSCHEMA, VIEWNAME
제한조건	CONSTSCHEMA, CONSTNAME
트리거	TRIGSCHEMA, TRIGNAME
패키지	PKGSCHEMA, PKGNAME
유형	TYPESCHEMA, TYPENAME, TYPEID
함수	FUNCSCHEMA, FUNCNAME, FUNCID
컬럼	COLNAME
속성	ATTR_NAME
스키마	SCHEMANAME
테이블 공간	TBSPACE
노드 그룹	NGNAME

버퍼 풀	BPNAME
이벤트 모니터	EVMONNAME
작성 시간소인	CREATE_TIME

카탈로그 뷰에 '로드맵'

설명	카탈로그 뷰	페이지
색인	OBJCAT.INDEXES	1392
색인 탐색 규칙	OBJCAT.INDEXEXPLOITRULES	1395
확장 종속성 색인	OBJCAT.INDEXEXTENSIONDEP	1396
확장 메소드 색인	OBJCAT. INDEXEXTENSIONMETHODS	1397
확장 매개변수 색인	OBJCAT.INDEXEXTENSIONPARMS	1398
확장 색인	OBJCAT.INDEXEXTENSIONS	1399
술어 스펙	OBJCAT.PREDICATESPECS	1400
변환	OBJCAT.TRANSFORMS	1401

OBJCAT.INDEXES

표 108. OBJCAT.INDEXES 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
INDSCHEMA	VARCHAR(128)		색인 이름
INDNAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		색인을 작성한 사용자
TABSCHEMA	VARCHAR(128)		색인이 정의되는 테이블이나 별명의 규정화된 이름
TABNAME	VARCHAR(128)		
COLNAMES	VARCHAR(640)		오름차순 또는 내림차순을 나타내기 위해 각각의 앞에 + 또는 -가 오는 컬럼 이름 목록. 경고: 이 컬럼은 나중에 제거될 것입니다. 이 정보에 대해서는 1323 페이지의 『SYSCAT.INDEXCOLUSE』를 사용하십시오.
UNIQUERULE	CHAR(1)		고유 규칙: D = 중복 허용 P = 1차 색인 U = 허용된 고유 항목만
MADE_UNIQUE	CHAR(1)		Y = 색인이 원래 고유하지 않으나 고유 또는 기본 키 제한조건을 지원하기 위해 고유 색인으로 변환됩니다. 제한조건이 삭제될 경우 색인은 다시 고유하지 않은 상태로 될 것입니다. N = 색인이 작성된 대로 유지됨
COLCOUNT	SMALLINT		키 컬럼 수 + 포함 컬럼 수(있는 경우)
UNIQUE_COLCOUNT	SMALLINT		고유 키에 필요한 컬럼 수. 항상 COLCOUNT보다 작거나 같아야 합니다. 포함 컬럼이 있는 경우에만 COLCOUNT보다 작습니다. 색인에 고유 키(중복 허용)가 없는 경우에는 -1입니다.
INDEXTYPE	CHAR(4)		색인 유형 CLUS = 클러스터링 REG = 정규

표 108. OBJCAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
ENTRYTYPE	CHAR(1)		H = 계층 테이블의 색인(H 테이블) L = 유형화 테이블의 논리 색인 유형화되지 않은 테이블상의 색인인 경우에는 공백
PCTFREE	SMALLINT		처음 색인을 구축하는 동안 예약될 각 색인 리프 페이지의 비율. 이 공간은 색인이 만들어진 후에 삽입될 경우를 대비하여 제공됩니다.
IID	SMALLINT		내부 색인 ID
NLEAF	INTEGER		리프 페이지의 수. 통계를 내지 않았을 경우에는 -1
NLEVELS	SMALLINT		색인 레벨 수. 통계를 내지 않았을 경우에는 -1
FIRSTKEYCARD	BIGINT		구별되는 첫번째 키 값의 수(통계를 내지 않았을 경우에는 -1).
FIRST2KEYCARD	BIGINT		색인의 처음 두 컬럼을 사용하는 구별 키의 수 (통계를 내지 않았거나 적용 불가능한 경우에는 -1).
FIRST3KEYCARD	BIGINT		색인의 처음 세 컬럼을 사용하는 구별 키의 수 (통계를 내지 않았거나 적용 불가능한 경우에는 -1).
FIRST4KEYCARD	BIGINT		색인의 처음 네 컬럼을 사용하는 구별 키의 수 (통계를 내지 않았거나 적용 불가능한 경우에는 -1).
FULLKEYCARD	BIGINT		구별되는 전체 키 값의 수. 통계를 내지 않았을 경우에는 -1.
CLUSTERRATIO	SMALLINT		색인의 데이터 클러스터링 등급. 통계를 내지 않았거나 세부사항 색인 통계를 낸 경우에는 -1(이 경우에는 CLUSTERFACTOR가 대신 사용됩니다).
CLUSTERFACTOR	DOUBLE		클러스터링 등급의 더 정교한 측정 또는 세부사항 색인 통계가 수집되지 않았거나 색인이 별명으로 정의되는 경우에는 -1.
SEQUENTIAL_PAGES	INTEGER		사이에 큰 간격이 없거나 거의 없을 경우 색인 키 순서대로 디스크에 지정된 리프 페이지의 수(통계가 사용 불가능한 경우에는 -1).
DENSITY	INTEGER		색인이 차지하는 페이지 범위에서 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로서, 퍼센트로 표시됩니다(0에서 100까지의 정수로 표시되고, 사용 가능한 통계가 없을 경우에는 -1).

표 108. OBJCAT.INDEXES 카탈로그 뷰 (계속)

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
USER_DEFINED	SMALLINT		이 색인이 사용자에 의해 정의되었고 삭제되지 않은 경우에는 1이고, 그렇지 않으면 0입니다.
SYSTEM_REQUIRED	SMALLINT		기본 또는 고유 키 제한조건에 이 색인이 필요하거나 이 색인이 입력된 테이블의 오브젝트 식별자(OID) 컬럼에 있는 색인인 경우에는 1입니다. 기본 키 또는 고유 키 제한조건에 이 색인이 필요하고 이 색인이 입력된 테이블의 오브젝트 식별자(OID) 컬럼에 있는 색인인 경우에는 2입니다. 그렇지 않으면 0입니다.
CREATE_TIME	TIMESTAMP		색인이 작성된 시간
STATS_TIME	TIMESTAMP	예	이 색인에 대해 기록된 통계를 변경한 최종 시간. 사용 가능한 통계가 없을 경우에는 널(NULL)입니다.
PAGE_FETCH_PAIRS	VARCHAR(254)		문자 양식에서 표현된, 쌍으로 된 정수의 목록. 각 쌍은 가설 버퍼에 있는 페이지 수를 나타내며, 페이지 패치의 수는 가설 버퍼를 사용하여 이 색인으로 테이블을 스캔하는 데 필요합니다(사용 가능한 데이터가 없을 경우 문자열 길이 0).
MINPCTUSED	SMALLINT		0이 아니라면, 온라인 색인 재구성이 작동 가능화되며 값은 페이지 병합 전의 최소 사용 공간의 임계값입니다.
REVERSE_SCANS	CHAR(1)		Y = 색인이 역 스캔을 지원함 N = 색인이 역 스캔을 지원하지 않음
INTERNAL_FORMAT	SMALLINT		색인의 내부 표현을 코드화합니다.
IESCHEMA	VARCHAR(128)	예	확장 색인의 규정화 이름. 일반 색인의 경우에는 널(NULL).
IENAME	VARCHAR(18)	예	
IEARGUMENTS	CLOB(32K)	예	색인 작성시 지정된 매개변수의 외부 정보. 일반 색인의 경우에는 널(NULL).
REMARKS	VARCHAR(254)	예	사용자가 제공한 주석 또는 널(NULL).

OBJCAT.INDEXEXPLOITRULES

각 행은 색인 탐색을 나타냅니다.

표 109. OBJCAT.INDEXEXPLOITRULES 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
FUNCID	INTEGER		함수 ID.
SPECID	SMALLINT		CREATE FUNCTION 문의 술어 스펙 수.
IESCHEMA	VARCHAR(128)		확장 색인의 규정화 이름
IENAME	VARCHAR(18)		
RULEID	SMALLINT		고유 탐색 규칙 ID.
SEARCHMETHODID	SMALLINT		특정 색인 확장의 검색 메소드 ID.
SEARCHKEY	VARCHAR(320)		색인 탐색에 사용되는 키.
SEARCHARGUMENT	VARCHAR(1800)		색인 탐색에 사용되는 검색 인수.

OBJCAT.INDEXEXTENSIONDEP

색인 확장이 다양한 데이터베이스 오브젝트에 대해 가지는 각각의 종속성 행을 포함합니다.

표 110. OBJCAT.INDEXEXTENSIONDEP 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
IESCHEMA	VARCHAR(128)		다른 오브젝트에 대해 종속성을 가지는 확장 색인의 규정화된 이름.
IENAME	VARCHAR(18)		
BTYPE	CHAR(1)		확장 색인이 종속된 오브젝트의 유형. A = 별명 F = 함수 인스턴스 또는 메소드 인스턴스 J = 서버 정의 O = 계층 SELECT 특권에 대한 "외부" 종속성 R = 구조화 유형 S = 요약 테이블 T = 테이블(유형화되지 않은) U = 유형화 테이블 V = 뷰(유형화되지 않은) W = 유형화 뷰 X = 확장 색인
BSCHEMA	VARCHAR(128)		색인 확장에 종속되는 오브젝트의 규정 이름 (BTYPE='F'인 경우에는 함수의 고유 이름입니다).
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	예	BTYPE='O', 'T', 'U', 'V' 또는 'W'인 경우, 종속 트리거에 필요한 테이블(또는 뷰)에 대한 특권을 코드화합니다. 그렇지 않으면 널(NULL).

OBJCAT.INDEXEXTENSIONMETHODS

각 행은 검색 방법을 나타냅니다. 한 개의 확장 색인이 여러 개의 검색 방법을 포함해도 됩니다.

표 111. OBJCAT.INDEXEXTENSIONMETHODS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
METHODNAME	VARCHAR(18)		검색 방법 이름
METHODID	SMALLINT		색인 확장의 메소드 수.
IESCHEMA	VARCHAR(128)		확장 색인의 규정화 이름
IENAME	VARCHAR(18)		
RANGEFUNCSCHEMA	VARCHAR(128)		범위 함수의 규정 이름
RANGEFUNCNAME	VARCHAR(18)		
RANGESPECIFICNAME	VARCHAR(18)		범위 함수 고유 이름.
FILTERFUNCSCHEMA	VARCHAR(128)		필터 기능의 규정화 이름
FILTERFUNCNAME	VARCHAR(18)		
FILTERSPECIFICNAME	VARCHAR(18)		필터 기능의 규정화 이름
REMARKS	VARCHAR(254)	예	사용자 제공 또는 널(NULL)

OBJCAT.INDEXEXTENSIONPARMS

각 행은 색인 확장 인스턴스 매개변수 또는 소스 키 정의를 나타냅니다.

표 112. OBJCAT.INDEXEXTENSIONPARMS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
IESCHEMA	VARCHAR(128)		확장 색인의 규정화 이름
IENAME	VARCHAR(18)		
ORDINAL	SMALLINT		매개변수 또는 소스 키의 순차 번호
PARMNAME	VARCHAR(18)		매개변수 또는 소스 키의 이름
TYPESCHEMA	VARCHAR(128)		인스턴스 매개변수 또는 소스 키 데이터 유형의 규정화 이름
TYPENAME	VARCHAR(18)		
LENGTH	INTEGER		인스턴스 매개변수 또는 소스 키 데이터 유형의 길이
SCALE	SMALLINT		인스턴스 매개변수 또는 소스 키 데이터 유형의 스케일. 적용이 불가능할 경우에는 제로(0)입니다.
PARMTYPE	CHAR(1)		행에 표시되는 유형: P = 확장 색인 매개변수 K = 키 컬럼
CODEPAGE	SMALLINT		확장 색인 매개변수의 코드 페이지. 문자열 유형이 아닐 경우는 제로(0)입니다.

OBJCAT.INDEXEXTENSIONS

각 확장 색인에 대한 행을 포함합니다.

표 113. OBJCAT.INDEXEXTENSIONS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
IESCHEMA	VARCHAR(128)		확장 색인의 규정화 이름
IENAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		확장 색인이 정의된 상태에서 권한 부여 ID
CREATE_TIME	TIMESTAMP		색인 확장이 정의된 시간.
KEYGENFUNCSCHEMA	VARCHAR(128)		키 생성 함수의 규정화 이름
KEYGENFUNCNAME	VARCHAR(18)		
KEYGENSPECIFICNAME	VARCHAR(18)		키 생성 함수 지정 이름.
TEXT	CLOB(64K)		CREATE INDEX EXTENSION문의 전체 텍스트
REMARKS	VARCHAR(254)		사용자가 제공한 주석 또는 널(NULL).

OBJCAT.PREDICATESPECS

표 114. OBJCAT.PREDICATESPECS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
FUNCSCHEMA	VARCHAR(128)		함수의 규정화 이름
FUNCNAME	VARCHAR(18)		
SPECIFICNAME	VARCHAR(18)		함수 인스턴스 이름
FUNCID	INTEGER		함수 ID
SPECID	SMALLINT		술어 스펙의 ID
CONTEXTOP	CHAR(8)		비교 연산자는 내장 관계 연산자 중 하나입니다 (=,<,>=, etc.).
CONTEXTEXP	CLOB(32K)		상수 또는 SQL 표현식
FILTERTEXT	CLOB(32K)	예	데이터 필터 표현식의 텍스트

OBJCAT.TRANSFORMS

명명된 변환 그룹에 포함되는 사용자 정의 유형내에 각 변환 함수 유형 행을 포함합니다.

표 115. OBJCAT.TRANSFORMS 카탈로그 뷰

컬럼 이름	데이터 유형	널 (NULL) 입력 가능	설명
TYPEID	SMALLINT		SYSCAT.DATATYPES에서 정의된 상태의 내부 유형 ID
TYPESCHEMA	VARCHAR(128)		사용자 정의 제공 구조화 유형의 규정 이름
TYPENAME	VARCHAR(18)		
GROUPNAME	VARCHAR(18)		그룹 이름 변환
FUNCID	INTEGER	예	SYSCAT.FUNCTIONS에 정의된 대로 연관 변환 함수에 대한 내부 함수 ID. 내부 시스템 함수인 경우에만 NULL.
FUNCSCHEMA	VARCHAR(128)		관련 변환 함수의 규정 이름
FUNCNAME	VARCHAR(18)		
SPECIFICNAME	VARCHAR(18)		함수 특정(인스턴스) 이름.
TRANSFORMTYPE	VARCHAR(8)		'FROM SQL' = 변환 함수가 SQL로부터 구조화 유형을 변환합니다. 'TO SQL' = 변환 함수가 구조화 유형을 SQL로 변환합니다.
FORMAT	CHAR(1)		'U' = 사용자 정의
MAXLENGTH	INTEGER	예	FROM SQL 변환으로부터의 최대 출력 길이(바이트 수). TO SQL 변환의 경우 NULL.
ORIGIN	CHAR(1)		'I' = 아래로 계승된 유형 계층. 'U' = 사용자 정의
REMARKS	VARCHAR(254)	예	사용자 제공 주석 또는 널(NULL)

부록F. 연합 시스템

이 부록에서는 다음에 대해 기술합니다.

- DB2 연합 시스템을 형성하고 사용하기 위해 SQL문에서 정의될 수 있는 서버 유형.
- DB2 연합 시스템을 형성하고 사용하기 위해 SQL문에서 정의될 수 있는 옵션.
- 연합 서버에 의해 지원되는 데이터 유형과 데이터 소스에 의해 지원되는 데이터 유형간의 기본 맵핑.
- 통과 사용시 고려할 요인 및 준수할 제한사항

서버 유형

서버 유형은 서버가 어떤 종류의 데이터 소스를 표현할 지에 대해 나타냅니다. 서버 유형은 벤더, 용도 및 플랫폼에 따라 다릅니다. 지원 값은 사용 중인 랩퍼에 따라 다릅니다.

- **DRDA 랩퍼**

  DB2 계열

표 116. IBM DB2 Universal Database

서버 유형	데이터 소스
DB2/UDB	IBM DB2 Universal Database
DataJoiner	IBM DB2 DataJoiner V2.1 및 V2.1.1
DB2/6000	AIX용 IBM DB2
DB2/HPUX	HP-UX용 IBM DB2 V1.2
DB2/NT	Windows NT용 IBM DB2
DB2/EEE	IBM DB2 Enterprise-Extended Edition
DB2/SUN	Solaris용 IBM DB2 V1 및 V1.2
DB2/2	OS/2용 IBM DB2
DB2/LINUX	Linux용 IBM DB2
DB2/PTX	NUMA-Q용 IBM DB2
DB2/SCO	SCO Unixware용 IBM DB2

표 117. AS/400용 IBM DB2 Universal Database

서버 유형	데이터 소스
DB2/400	AS/400용 IBM DB2

표 118. OS/390용 IBM DB2 Universal Database

서버 유형	데이터 소스
DB2/390	OS/390용 IBM DB2
DB2/MVS	MVS용 IBM DB2

표 119. VM 및 VSE용 IBM DB2 서버

서버 유형	데이터 소스
DB2/VM	VM용 IBM DB2
DB2/VSE	VSE용 IBM DB2
SQL/DS	IBM SQL/DS

- **SQLNET 랩퍼**

  Oracle 데이터 소스는 Oracle SQL*Net V1 또는 V2 클라이언트 소프트웨어에서 지원합니다.

서버 유형	데이터 소스
ORACLE	Oracle V7.0.13 이상

- **NET8 랩퍼**

  Oracle 데이터 소스는 Oracle Net8 클라이언트 소프트웨어에서 지원합니다.

서버 유형	데이터 소스
ORACLE	Oracle V7.0.13 이상

- **OLE DB 랩퍼**

  Microsoft OLE DB 2.0 이상에 맞는 OLE DB 제공자.

서버 유형	데이터 소스
-	모든 OLE DB 제공자

- **기타 랩퍼**

  랩퍼와 함께 제공된 문서를 참조하십시오.

연합 시스템을 위한 SQL 옵션

이 절에서는 다음에 대해 기술합니다.

- ALTER NICKNAME문에서 지정될 수 있는 컬럼 옵션
- CREATE FUNCTION MAPPING문에서 지정될 수 있는 함수 맵핑 옵션
- CREATE SERVER, ALTER SERVER 및 SET SERVER OPTION문에서 지정될 수 있는 서버 옵션
- CREATE USER MAPPING 및 ALTER USER MAPPING문에서 지정될 수 있는 사용자 옵션

컬럼 옵션

컬럼 옵션의 1차적인 용도는 별명 컬럼에 대한 정보를 SQL 컴파일러에 제공하는 것입니다. 하나 이상이 컬럼에 대한 컬럼 옵션을 'Y'로 설정하면 컴파일러가 평가 조작을 수행하는 술어에 대한 추가 푸시 다운 가능성을 고려할 수 있게 합니다. 푸시 다운 처리에 대한 *관리 안내서: 성능*에서 자세한 정보를 참조하십시오.

표 120. 컬럼 옵션 및 그 설정값

옵션	유효한 설정값		기본 설정값
numeric_string	'Y'	이 컬럼이 숫자 데이터의 문자열만을 포함함을 의미하는 Yes. 중요: 컬럼이 공백이 뒤따르는 숫자 문자열만을 포함하는 경우에는 'Y'를 지정하지 않는 것이 좋습니다.	'N'
	'N'	컬럼이 숫자 데이터의 문자열에 재한됨을 의미하는 No.	
	컬럼에 대한 numeric_string을 'Y'로 설정하여, 최적화 알고리즘에게 이 컬럼이 컬럼의 데이터를 방해할 수도 있는 공백을 포함하지 않는다고 알립니다. 이 옵션은 데이터 소스의 조합 순서가 DB2와 다를 때에 유용합니다. 이 옵션으로 표시된 컬럼은 서로 다른 조합 순서 때문에 지역(데이터 소스) 평가에서 제외되지 않을 것입니다.		

표 120. 컬럼 옵션 및 그 설정값 (계속)

옵션	유효한 설정값	기본 설정값
varchar_no_trailing_blanks	특정 VARCHAR 컬럼에 뒤 공백이 없는지 여부를 나타냅니다. 'Y' 예, 이 VARCHAR 컬럼 뒤에 공백이 없음을 의미 'N' 아니오, 이 VARCHAR 컬럼 뒤에 공백이 없음을 의미 데이터 소스 VARCHAR 컬럼에 공백이 채워져 있지 않으면, 이들을 액세스하는 최적화 알고리즘의 전략은 이들이 뒤 공백을 포함하는지 여부에 따라 일부 달라집니다. 기본적으로, 최적화 알고리즘은 이들이 실제로 뒤 공백을 포함한다고『가정합니다』. 이러한 가정에서, 이들 컬럼에서 리턴된 값들이 사용자가 기대하는 것이 되도록 조회를 수정하는 일이 수반되는 액세스 전략을 개발합니다. 그러나, VARCHAR 컬럼에 뒤 공백이 없으며, 최적화 알고리즘이 이를 알게 하면, 보다 효율적인 액세스 전략을 개발할 수 있습니다. 최적화 알고리즘에게 특정 컬럼이 뒤 공백을 가지고 있지 않음을 알리려면, ALTER NICKNAME문에서 그 컬럼을 지정하십시오(구문에 대해서는 *SQL 참조서* 참조).	'N'

함수 맵핑 옵션

함수 맵핑 옵션의 1차적인 용도는 데이터 소스에서 데이터 소스 함수를 실행하는 잠재적인 비용에 대한 정보를 제공하는 것입니다. 푸시다운 분석이 맵핑 내의 두 함수 중 어느 하나라도 호출될 수 있는지를 결정하면, 맵핑 정보에서 제공된 통계 정보가 최적화 알고리즘으로 하여금 추정 데이터 소스 실행 비용을 추정 DB2 함수 실행 비용과 비교할 수 있게 돕습니다.

표 121. 함수 맵핑 옵션 및 그 설정값

옵션	유효한 설정값	기본 설정값
사용안함	기본 함수 맵핑을 사용안합니다. 유효한 값은 'Y'와 'N'입니다.	'N'
initial_insts	데이터 소스 함수가 처음 그리고 마지막에 호출될 때 처리된 예상 명령어 수.	'0'
initial_ios	데이터 소스 함수가 처음 그리고 마지막에 호출될 때 수행된 예상 I/O 수.	'0'
ios_per_argbyte	데이터 소스 함수에 전달된 인수 세트의 각 바이트에 소모된 예상 I/O 수.	'0'
ios_per_invoc	데이터 소스 함수의 호출당 I/O 평가 수.	'0'

표 121. 함수 맵핑 옵션 및 그 설정값 (계속)

옵션	유효한 설정값	기본 설정값
insts_per_argbyte	데이터 소스 함수에 전달된 인수 세트의 각 바이트에 처리된 예상 명령어 수.	'0'
insts_per_invoc	데이터 소스 함수의 각 호출 당 처리된 예상 명령어 수.	'450'
percent_argbytes	데이터 소스 함수가 실제로 읽을 입력 인수 바이트의 예상 평균 백분율.	'100'
remote_name	데이터 소스 함수의 이름.	지역 이름

서버 옵션

서버 옵션이 서버를 설명하는 데 사용됩니다. 위치 정보(데이터 소스 머신 이름과 같은)외에, 옵션은 데이터 소스에 대한 보안 및 성능 속성을 지정할 수 있습니다. 보안 옵션은 암호 통신에 대한 제어(데이터 소스에 보낼지 안 보낼지)와 인증 정보 케이스(대문자 및/또는 소문자 ID와 암호)를 제공합니다. 성능 옵션은 최적화 알고리즘으로 하여금 데이터 소스에서 평가 조작이 행해질 수 있는지와 지료 소스로부터 데이터를 검색하는 조회를 완료하기 위한 최상의 비용 모델을 결정할 수 있게 도와줍니다.

표 122. 서버 옵션 및 그 설정값

옵션	유효한 설정값	기본 설정값
collating_sequence	데이터 소스가 코드 세트 및 국가 정보에 기초하여 연합 데이터베이스와 동일한 기본 조합 순서를 사용하는지의 여부를 지정합니다. 데이터 소스가 DB2의 조합 순서와 다른 조합 순서를 가진다면, DB2의 조합 순서에 의존하는 대부분의 조작이 데이터 소스에서 원격으로 평가될 수 없습니다. 서로 다른 조합 순서를 가진 데이터 소스에서 별명 컬럼에 대해 MAX 컬럼 함수를 실행하는 것이 한 예입니다. 원격 데이터 소스에서 MAX 함수가 평가되면 결과가 다를 수도 있으므로, DB2는 총계 조적 및 MAX 함수를 지역적으로 수행할 것입니다. 조회에 등호가 있으면, 조합 순서가 달라져도('N'으로 설정) 조회의 그 부분을 푸시다운하는 것이 가능합니다. 예를 들어, 술어 C1 = 'A'는 데이터 소스로 푸시다운될 수 있습니다. 물론, 데이터 소스에 있는 조합 순서가 대소문자를 구별할 때에는 그러한 조회를 푸시다운할 수가 없습니다. 데이터 소스가 대소문자를 구별하면, C1= 'A' 및 C1 = 'a'로부터의 결과가 같으며, 이는 대소문자를 구별하는 환경(DB2)에서 허용되지 않습니다. 관리자는 데이터 소스 조합 순서와 일치하는 특정 조합 순서를 가진 연합 데이터베이스를 작성할 수 있습니다. 모든 데이터 소스가 동일한 조합 순서를 사용하거나 대부분의 또는 모든 컬럼 함수가 동일한 조합 순서를 사용하는 데이터 소스로 지정되는 경우에는 이러한 접근 방식이 성능을 높일 수도 있습니다. 'Y' 데이터 소스의 조합 순서가 연합 데이터베이스의 조합 순서와 같습니다. 'N' 데이터 소스의 조합 순서가 연합 데이터베이스의 조합 순서와 다릅니다. 'I' 데이터 소스의 소스가 연합 데이터베이스의 조합 순서와 다르며 대소문자를 구별하지 않습니다.(예를 들어, 'TOLLESON'과 'TolLESon'은 같은 것으로 여겨집니다.)	'N'
comm_rate	연합 서버와 연관 데이터 소스간의 통신을 지정합니다. 초당 메가바이트 단위로 표현됩니다. 유효한 값은 0보다 크거나 2147483648보다 작아야 합니다. 값은 숫자로만 표시될 수 있습니다(예: 12).	'2'

표 122. 서버 옵션 및 그 설정값 (계속)

옵션	유효한 설정값	기본 설정값
connectstring	OLE DB 공급자 연결하는 데 필요한 초기화 등록 정보를 지정합니다. 연결 문자열의 전체 구문 및 시멘틱에 대해서는, *Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK, Microsoft Press, 1998*에서 "Data Link API of the OLE DB Core Components"를 참조하십시오.	없음
cpu_ratio	데이터 소스의 CPU가 연합 서버의 CPU 보다 얼마나 더 빨리 또는 느리게 수행하는지 나타냅니다. 유효한 값은 0보다 크거나 2^{3}보다 작아야 합니다. 값은 유효한 DOUBLE 표기법으로 표시할 수 있습니다(예: 123E10, 123 또는 1.21E4).	'1.0'
dbname	연합 데이터베이스가 액세스하게 하고 싶은 데이터 소스 데이터베이스의 이름. DB2 계열 데이터 소스에 필요합니다. Oracle 인스턴스에는 하나의 데이터베이스만 포함되므로 Oracle** 데이터 소스에는 적용하지 마십시오. DB2의 경우 이 값은 하나의 인스턴스에 있는 특정 데이터베이스에 해당되며, OS/390용 DB2의 경우에는 데이터베이스 LOCATION 값입니다.	없음.
fold_id (이 테이블 끝에 있는 주 1과 4 참조).	연합 서버가 인증을 위해 데이터 소스에 전송하는 사용자 ID에 적용됩니다. 가능한 값은 다음과 같습니다. 'U' 연합 서버는 데이터 소스에 전송하기에 앞서 사용자 ID를 대문자로 합니다. DB2 계열 및 Oracle** 데이터 소스용 논리 선택사항입니다(이 테이블 끝에 있는 주 2 참조). 'N' 연합 서버는 데이터 소스에 전송하기에 앞서 사용자 ID에 대해 아무것도 하지 않습니다(이 테이블 끝에 있는 주 2 참조). 'L' 연합 서버는 데이터 소스에 전송하기에 앞서 사용자 ID를 소문자로 합니다. 이들 설정값 중 어느것도 사용되지 않으면, 연합 서버가 대문자로 사용자 ID를 데이터 소스에 전송하려고 시도합니다. 사용자 ID가 실패하면, 서버가 이를 소문자로 전송하려고 시도합니다.	없음.

표 122. 서버 옵션 및 그 설정값 (계속)

옵션	유효한 설정값	기본 설정값
fold_pw (이 테이블 끝에 있는 주 1, 3 및 4 참조).	연합 서버가 인증을 위해 데이터 소스에 전송하는 암호에 적용됩니다. 가능한 값: 'U' 연합 서버는 데이터 소스에 전송하기에 앞서 암호를 대문자로 합니다. DB2 계열 및 Oracle** 데이터 소스용 논리 선택사항입니다. 'N' 연합 서버는 데이터 소스에 전송하기에 앞서 암호에 대해 아무것도 하지 않습니다 'L' 연합 서버는 데이터 소스에 전송하기에 앞서 암호를 소문자로 합니다. 이들 설정값 중 어느것도 사용되지 않으면, 연합 서버가 대문자로 암호를 데이터 소스에 전송하려고 시도합니다. 암호가 실패하면, 서버가 이를 소문자로 전송하려고 시도합니다.	없음.
io_ratio	데이터 소스의 입출력 시스템이 연합 서버의 입출력 시스템보다 얼마나 더 빨리 또는 느리게 수행하는지 나타냅니다. 유효한 값은 0보다 크거나 ^{23}보다 작아야 합니다. 값은 유효한 DOUBLE 표기법으로 표시할 수 있습니다(예: 123E10, 123 또는 1.21E4).	'1.0'
노드	데이터 소스가 RDBMS에 대한 인스턴스로서 정의되는 이름. 모든 데이터 소스에 필요합니다. DB2 계열 데이터 소스의 경우, 이 이름은 연합 데이터베이스의 DB2 노드 디렉토리에서 지정된 노드입니다. 이 디렉토리를 보려면 **db2 list node directory** 명령을 발행하십시오. Oracle** 데이터 소스의 경우, 이 이름은 Oracle** tnsnames.ora 파일에서 지정된 서버 이름입니다. Windows NT 플랫폼에서 이 이름을 액세스하려면, Oracle** SQL Net 간편 구성 도구의 **구성 정보 보기** 옵션을 지정하십시오.	없음.

표 122. 서버 옵션 및 그 설정값 (계속)

옵션	유효한 설정값	기본 설정값
암호	암호가 데이터 소스에 전송되는지 여부를 지정합니다. 'Y' 암호가 항상 데이터 소스에 전송되고 유효화됩니다. 이것은 기본 값입니다. 'N' 암호가 항상 데이터 소스에 전송되고(사용자 맵핑에 관계없이) 유효화되지 않습니다. 'ENCRYPTION' 암호는 항상 암호화 양식으로 데이터 소스에 전송되고 유효성 확인됩니다. 암호화된 암호를 지원하는 DB2 계열 데이터 소스에만 유효합니다.	'Y'
plan_hints	플랜 힌트가 작동가능화 되는지의 여부를 지정합니다. 플랜 힌트는 데이터 소스 최적화 알고리즘을 위한 추가 정보를 제공하는 명령문 단편입니다. 특정 조회 유형의 경우에 이 정보가 조회 성능을 향상시킬 수 있습니다. 플랜 힌트는 데이터 소스 최적화 알고리즘이 색인을 사용할지 말지, 어느 색인을 사용할지 또는 어느 테이블 조인 순서를 사용할지 결정하는 것을 돕습니다. 'Y' 데이터 소스가 플랜 힌트를 지원하면 플랜 힌트가 데이터 소스에서 작동가능화 될 것입니다. 'N' 플랜 힌트가 데이터 소스에서 작동가능화되지 않을 것입니다.	'N'
pushdown	'Y' DB2가 데이터 소스 평가 조작을 고려할 것입니다. 'N' DB2가 원격 데이터 소스로부터의 컬럼만 검색할 것이며, 데이터 소스가 조인과 같은 다른 조작을 평가하지 않게 할 것입니다.	'Y'

표 122. 서버 옵션 및 그 설정값 (계속)

옵션	유효한 설정값	기본 설정값
varchar_no_trailing_blanks	이 데이터 소스가 공백으로 채워지지 않은 비교 시멘틱을 사용하는지 여부를 지정합니다. 뒤 공백을 포함하지 않는 가변 길이 문자열의 경우, 일부 DBMS의 공백으로 채워지지 않은 비교 시멘틱이 DB2의 비교 시멘틱과 동일한 결과를 리턴합니다. 데이터 소스에 있는 모든 VARCHAR 테이블/뷰 컬럼이 뒤 공백을 포함하고 있지 않다고 확신하면, 데이터 소스에 대해 이 서버 옵션을 'Y'로 설정할 것을 고려하십시오. 이 옵션은 종종 Oracle** 데이터 소스와 사용됩니다. 잠재적으로 별명(뷰를 포함하여)을 가질 수 있는 모든 오브젝트를 고려하십시오. 'Y' 이 데이터 소스에 DB2와 유사한 공백으로 채워지지 않은 비교 시멘틱이 있습니다. 'N' 이 데이터 소스에 DB2와 유사한 공백으로 채워지지 않은 비교 시멘틱이 없습니다.	'N'

1408 페이지의 표122에 대한 참고사항:

1. 인증에 지정된 값에 관계없이 이 필드가 적용됩니다.
2. DB2가 사용자 ID를 대문자로 보관하기 때문에, 'N' 및 'U' 값이 논리적으로 서로 서로에 해당합니다.
3. 암호를 'N'으로 설정할 때에는 fold_pw에 대한 설정값이 아무런 효과도 내지 못합니다. 암호가 전송되지 않기 때문에, 케이스가 인수가 될 수 없습니다.
4. 이들 옵션들에 대해서는 널(NULL) 설정값을 피하십시오. DB2가 사용자 ID 및 암호를 해석하기 위해 여러 시도를 할 것이기 때문에 널(NULL) 설정값에 매력을 느낄 수도 있습니다. 그러나 성능이 저하될 수도 있습니다(DB2가 성공적으로 데이터 소스 인증을 전달하기 전에 사용자 ID 및 암호를 네번 전송할 가능성이 있으므로).

사용자 옵션

사용자 옵션은 사용자 맵핑을 위한 권한 부여 및 사용통계 문자열 정보를 제공합니다. 이들을 사용하여 데이터 소스에서 인증시 DB2 인증 ID를 표현하기 위해 사용되는 ID 및 암호를 지정하십시오.

표 123. 사용자 옵션 및 그 설정값

옵션	유효한 설정값	기본 설정값
remote_authid	데이터 소스에서 사용된 권한 부여 ID를 나타냅니다. 유효한 설정값은 255 이하의 임의의 문자열을 포함합니다. 이 옵션을 지정하지 않으면, 데이터베이스에 연결하는 데 사용된 ID가 사용됩니다.	없음.
remote_domain	이러한 데이터 소스에 연결되어 있는 사용자를 인증하는 데 사용되는 Windows NT 도메인을 나타냅니다. 유효한 설정값은 모든 유효한 Windows NT 도메인 이름을 포함합니다. 이 옵션을 지정하지 않으면, 데이터 소스는 해당 데이터베이스에 대한 인증 도메인 기본값으로 인증됩니다.	없음.
remote_password	데이터 소스에서 사용된 권한 부여 암호를 나타냅니다. 유효한 설정값은 32 이하의 임의의 문자열을 포함합니다. 이 옵션을 지정하지 않으면, 데이터베이스에 연결하는 데 사용된 암호가 사용됩니다.	없음.
accounting_string	DRDA 사용통계 문자열을 지정하는 데 사용됩니다. 유효한 설정값은 255 이하의 임의의 문자열을 포함합니다. 이 옵션은 사용통계 정보가 전달되어야 하는 경우에만 필요합니다. *DB2 Connect 사용자 안내서*에서 참조하십시오.	없음.

기본 데이터 유형 맵핑

이 절에서는 연합 서버에 의해 지원되는 DB2 데이터 유형과 다음의 데이터 소스에 의해 지원되는 데이터 유형간의 기본 맵핑을 보여줍니다.

- OS/390용 DB2 Universal Database 및 MVS/ESA용 DB2
- AS/400용 DB2 Universal Database 및 OS/400용 DB2
- Oracle
- VM용 DB2 및 VSE; SQL/DS

표시된 맵핑은 동일하지 않은 데이터 유형간의 맵핑입니다. 동일한 데이터 유형간의 맵핑은 보여주지 않습니다.

DB2와 OS/390용 DB2 Universal Database(및 MVS/ESA용 DB2) 데이터 소스간의 기본 맵핑

표 124. DB2와 OS/390용 DB2 Universal Database(및 MVS/ESA용 DB2) 데이터 소스간의 기본 맵핑

MVS용 DB2, OS/390용 DB2	DB2
varchar(n), n <= 32672	varchar(n)
vargraphic(n), n <= 16336	vargraphic(n)
char(255)	varchar(255)
2진 데이터용에 대한 char(255)	2진 데이터용에 대한 varchar(255)

DB2와 AS/400용 DB2 Universal Database(및 OS/400용 DB2) 데이터 소스간의 기본 유형 맵핑

표 125. DB2와 AS/400용 DB2 Universal Database(및 OS/400용 DB2) 데이터 소스간의 기본 유형 맵핑

OS/400용 DB2, AS/400용 DB2	DB2
char(n), n은 <= 254임	char(n)
char(n), n은 255와 32672 사이임	varchar(n)
varchar(n), n은 <= 32672임	varchar(n)
graphic(n), n은 <= 127임	graphic(n)
graphic(n), n은 127와 16336 사이임	vargraphic(n)
vargraphic(n), n은 <= 16336임	vargraphic(n)

DB2와 Oracle 데이터 소스간의 기본 유형 맵핑

표 126. DB2와 Oracle 데이터 소스간의 기본 유형 맵핑

Oracle	DB2
rowid	char(18)
char(n), n은 <= 254임	char(n)
nchar(n), n은 <= 254임	char(n)
char(255)	varchar(255)
varchar2(n), n은 <= 32672임	varchar(n)
nvarchar2(n), n은 <= 32672임	varchar(n)
number(p,s), p는 <= 4이고 s = 0임	smallint

표 126. DB2와 Oracle 데이터 소스간의 기본 유형 맵핑 (계속)

Oracle	DB2
number(p,s), 4 <= p이고 <= 9 및 s = 0임	정수
number(p,s), 10 <= p이고 <= 18 및 s = 0임	bigint
number(p,s), p <= 31 및 0 <= s <= p이고 앞의 두 경우가 일치하지 않음	decimal
number(p,s), 앞의 네 경우가 아닌 다른 모든 경우	double
raw(n), n은 <= 254임	2진 데이터용 char(n)
raw(255)	2진 데이터용 varchar(255)
날짜 (char(9))	시간소인

DB2와 VM 및 VSE용 DB2(및 SQL/DS) 데이터 소스간의 기본 유형 맵핑

표 127. DB2와 VM 및 VSE용 DB2(및 SQL/DS) 데이터 소스간의 기본 유형 맵핑

OS/390용 DB2, SQL/DS	DB2
varchar(n), n은 <= 32672임	varchar(n)
vargraphic(n), n은 <= 16336임	vargraphic(n)

통과 기능 처리

*통과*라 불리는 기능을 사용하여 해당 데이터 소스에 원시인 SQL에서 데이터 소스를 조회할 수 있습니다. 이 절에서는 다음과 같습니다.

- 연합 서버 및 그 연관 데이터 소스가 통과 세션에서 어떤 종류의 SQL문을 처리하는지에 대해 기술합니다.
- 통과 사용시 주의할 고려사항 및 제한사항을 나열합니다.

통과 세션에서의 SQL 처리

다음 규칙은 SQL문이 DB2에 의해 처리되는지 또는 데이터 소스에 의해 처리되는지를 지정합니다.

- 통과 세션 처리를 위해 SQL문이 데이터 소스에 제출되는 경우, 이 SQL문은 세션에서 동적으로 준비되어 세션이 아직 열려 있는 상태에서 실행되어야 합니다. 이와 같이 하는 방법에는 몇 가지가 있습니다.
  - SELECT 문이 제출되면, PREPARE 문을 사용하여 준비한 다음 OPEN, FETCH 및 CLOSE 문을 사용하여 조회 결과에 액세스하십시오.
  - SELECT가 아닌 지원되는 다른 명령문의 경우,
    - PREPARE 문을 사용하여 지원되는 명령문을 준비한 다음 EXECUTE 문을 사용하여 실행하거나
    - EXECUTE IMMEDIATE문을 사용하여 준비하고 실행하십시오.
- 통과 세션에서 정적 명령문이 제출되면, 처리를 위해 연합 서버에 전송됩니다.
- COMMIT 또는 ROLLBACK 명령이 통과 세션 동안 발행되면, 이 명령이 현재 작업 단위(UOW)를 완료할 것입니다.

고려사항 및 제한사항

통과에 적용하는 많은 고려사항 및 제한사항이 있습니다. 이들 중 일부는 일반적인 성질의 것이며, 다른 것들은 Oracle 데이터 소스에만 유의합니다.

모든 데이터 소스와 통과 사용

다음 정보는 모든 데이터 소스에 적용합니다.

- 통과 세션내에서 준비된 명령문은 같은 통과 세션내에서 실행해야 합니다. 통과 세션내에서 준비되었으나 같은 통과 세션 외부에서 실행된 명령문은 실패합니다 (SQLSTATE 56098).
- 사용자와 응용프로그램은 통과를 사용하여 가령, 테이블 행을 삽입, 갱신 및 삭제하기 위해 데이터 소스 쓰기가 가능합니다. 통과 세션에서는 데이터 소스 오브젝트에 대해 커서가 직접 열릴 수 없다는 것을 유념하십시오.
- 응용프로그램은 서로 다른 데이터 소스에 여러 개의 SET PASSTHRU문이 동시에 영향을 미치게 할 수 있습니다. 응용프로그램이 여러 개의 SET PASSTHRU문을 발행했을 지라도, 통과 세션들이 정말 중첩되지는 않습니다. 연합 서버는 다른 것을 액세스하기 위해 한 데이터 소스를 통해 전달하지 않을 것입니다. 대신, 서버는 각 데이터 소스를 직접 액세스합니다.

- 여러 개의 통과 세션이 동시에 열리면, 각 세션 내의 각 작업 단위(UOW)가 COMMIT 또는 ROLLBACK문으로 종결되어야 합니다. 그런 후 SET PASSTHRU문 및 RESET 옵션으로 한 조작으로 세션이 종료될 수 있습니다.
- 한 번에 두 개 이상의 데이터 소스를 통과하는 것은 불가능합니다.
- 통과는 저장 프로시듀어 호출을 지원하지 않습니다.
- 통과는 SELECT INTO문을 지원하지 않습니다.

Oracle 데이터 소스와 통과 사용

다음 정보는 Oracle 데이터 소스에 적용합니다.

- 다음 제한사항은 원격 클라이언트가 통과 모드로 명령행 프로세서(CLP)에서 SELECT 문을 발행하는 경우에 적용됩니다. 클라이언트 코드가 DB2 Universal Database 버전 5 이전의 DB2 SDK인 경우에는 SELECT 문이 SQLSTATE 25000을 발생시킵니다. 이 오류를 피하기 위해서는, 원격 클라이언트가 버전 5 이상에서 DB2 SDK를 사용해야 합니다.
- Oracle 서버에 대해 발행된 임의의 DDL문은 분석시에 수행되며 트랜잭션 시멘틱에 따라 달라지지 않습니다. 조작은 완료시 Oracle에 의해 자동으로 확약됩니다. 구간 복원이 발생하면, DDL이 구간 복원되지 않습니다.
- 원시 데이터 유형으로부터 SELECT문이 발행될 때에는, 16진수 값을 수신하기 위해 RAWTOHEX 함수를 호출해야 합니다. 원시 데이터 유형으로의 INSERT가 수행될 때에는, 16진수 표현을 제공해야 합니다.

부록G. 샘플 데이터베이스 테이블

이 부록은 샘플 데이터베이스 SAMPLE의 샘플 테이블에 포함된 정보와 샘플 테이블을 작성하고 제거하는 방법을 보여줍니다.

비지니스 인텔리전스 함수를 설명하고 비지니스 인텔리전스 자습서에서 사용되는 추가 샘플 데이터베이스가 DB2 Universal Database에 제공됩니다. 그러나 이 부록에서는 샘플 데이터베이스 SAMPLE의 내용만 설명됩니다. 비지니스 인텔리전스 샘플 데이터베이스에 대해 자세히 알려면 *Data Warehouse Center 관리 안내서*를 참조하십시오.

샘플 테이블은 이 라이브러리에 있는 다른 지침서와 이 지침서에 나타난 예에서 사용됩니다. 또한, CLOB 데이터 유형 및 BLOB로 된 샘플 파일에 포함된 데이터를 나타냅니다.

다음 절은 이 부록에 포함되어 있습니다.

1434 페이지의『BLOB 및 CLOB 데이터 유형으로 된 샘플 파일』
1434 페이지의『Quintana 사진』
1434 페이지의『Quintana 이력서』
1436 페이지의『Nicholls 사진』
1436 페이지의『Nicholls 이력서』
1438 페이지의『Adamson 사진』
1438 페이지의『Adamson 이력서』
1440 페이지의『Walker 사진』
1440 페이지의『Walker 이력서』.

샘플 테이블에서, 대쉬(-)는 널(NULL) 값을 나타냅니다.

샘플 데이터베이스

이 책의 예에서는 샘플 데이터베이스를 사용합니다. 이들 예를 사용하려면, SAMPLE 데이터베이스를 작성해야 합니다. 샘플 테이타베이스를 사용하려면, 데이터베이스 관리 프로그램을 설치하십시오.

샘플 데이터베이스 작성하기

실행 파일은 샘플 데이터베이스를 작성합니다.[116] 데이터베이스를 작성하려면 SYSADM 권한이 있어야 합니다.

- **UNIX 기반 플랫폼 사용시**

  운영 체제 명령 프롬프트를 사용할 경우, 다음을 입력하십시오.

  ```
  sqllib/bin/db2sampl <path>
  ```

  데이터베이스 관리 프로그램 인스턴스 소유자의 홈 디렉토리에서, *path*는 동일한 데이터베이스를 작성할 곳의 경로를 지정하는 선택적 매개변수입니다. Enter를 누르십시오.[117] DB2SAMPL용 스키마는 CURRENT SCHEMA 특수 레지스터 값입니다.

116. 이 명령과 관련된 정보에 대해서는 *Command Reference*의 DB2SAMPL 명령을 참조하십시오.

117. 경로 매개변수가 지정되지 않은 경우, 샘플 데이터베이스는 데이터베이스 관리 프로그램 구성 파일의 DFTDBPATH 매개변수에 의해 지정된 기본 경로에 작성됩니다.

- **OS/2 또는 Windows 플랫폼 사용시**

  운영 체제 명령 프롬프트를 사용할 경우, 다음을 입력하십시오.

  ```
  db2sampl e
  ```

  여기서, *e*는 데이터베이스가 작성될 곳의 드라이브를 지정하는 선택적 매개변수입니다. Enter를 누르십시오.[118]

  사용자 프로파일 관리를 통해서 워크스테이션에 로그되어 있지 않으면, 로그하도록 명령합니다.

샘플 데이터베이스 지우기

샘플 데이터베이스 액세스가 필요 없을 경우, DROP DATABASE 명령을 사용하여 작성할 수 있습니다.

```
db2 drop database sample
```

CL_SCHED 테이블

이름:	CLASS_CODE	DAY	STARTING	ENDING
유형:	char(7)	smallint	시간	시간
Desc:	클래스 코드 (room:teacher)	4일 스케줄의 날짜 #	클래스 시작 시간	클래스 종료 시간

DEPARTMENT 테이블

이름:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
유형:	char(3) 널(NULL)이 아님	varchar(29) 널(NULL)이 아님	char(6)	char(3) 널(NULL)이 아님	char(16)
Desc:	부서 번호	일반적인 부서 업무를 설명하는 이름	부서 관리자의 사원 번호 (EMPNO)	이 부서에서 보고할 부서 (DEPTNO)	원격 위치 이름
값:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	-
	B01	PLANNING	000020	A00	-

118. 드라이브 매개변수가 지정되지 않은 경우, 샘플 데이터베이스는 DB2와 같은 드라이브에 작성됩니다.

이름:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
	C01	INFORMATION CENTER	000030	A00	-
	D01	DEVELOPMENT CENTER	-	A00	-
	D11	MANUFACTURING SYSTEMS	000060	D01	-
	D21	ADMINISTRATION SYSTEMS	000070	D01	-
	E01	SUPPORT SERVICES	000050	A00	-
	E11	OPERATIONS	000090	E01	-
	E21	SOFTWARE SUPPORT	000100	E01	-

EMPLOYEE 테이블

이름:	EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
유형:	char(6) 널(NULL)이 아님	varchar(12) 널(NULL)이 아님	char(1) 널(NULL)이 아님	varchar(15) 널(NULL)이 아님	char(3)	char(4)	날짜 시간
Desc:	사원 번호	이름	중간 이름 첫 자	성	사원이 속한 부서 (DEPTNO)	전화 번호	입사 일자

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(8)	smallint 널(NULL)이 아님	char(1)	날짜 시간	dec(9,2)	dec(9,2)	dec(9,2)
작업	정식 교육을 받은 연수	성별(M 남성, F 여성)	생년 월일	연간 봉급	연간 보너스	연간 커미션

EMPLOYEE 테이블의 값에 대해서는 다음 페이지를 참조하십시오.

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(6) 널 (NULL)이 아님	varchar(12) 널 (NULL)이 아님	char(1) 널 (NULL)이 아님	varchar(15) 널 (NULL)이 아님	char(3)	char(4)	날짜 시간	char(8)	smallint 널 (NULL)이 아님	char(1)	날짜	dec(9,2)	dec(9,2)	dec(9,2)
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950	400	1596
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370	500	2030
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840	500	1907

EMP_ACT 테이블

이름:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
유형:	char(6) 널 (NULL)이 아님	char(6) 널 (NULL)이 아님	smallint 널 (NULL)이 아님	dec(5,2)	날짜	날짜
Desc:	사원 번호	프로젝트 번호	작업 번호	프로젝트에 보낸 사원 시간의 비율	작업 시작일	작업 종료일
값:	000010	AD3100	10	.50	1982-01-01	1982-07-01
	000070	AD3110	10	1.00	1982-01-01	1983-02-01
	000230	AD3111	60	1.00	1982-01-01	1982-03-15
	000230	AD3111	60	.50	1982-03-15	1982-04-15
	000230	AD3111	70	.50	1982-03-15	1982-10-15
	000230	AD3111	80	.50	1982-04-15	1982-10-15
	000230	AD3111	180	1.00	1982-10-15	1983-01-01
	000240	AD3111	70	1.00	1982-02-15	1982-09-15
	000240	AD3111	80	1.00	1982-09-15	1983-01-01
	000250	AD3112	60	1.00	1982-01-01	1982-02-01
	000250	AD3112	60	.50	1982-02-01	1982-03-15
	000250	AD3112	60	.50	1982-12-01	1983-01-01
	000250	AD3112	60	1.00	1983-01-01	1983-02-01
	000250	AD3112	70	.50	1982-02-01	1982-03-15
	000250	AD3112	70	1.00	1982-03-15	1982-08-15
	000250	AD3112	70	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.50	1982-10-15	1982-12-01
	000250	AD3112	180	.50	1982-08-15	1983-01-01
	000260	AD3113	70	.50	1982-06-15	1982-07-01
	000260	AD3113	70	1.00	1982-07-01	1983-02-01
	000260	AD3113	80	1.00	1982-01-01	1982-03-01
	000260	AD3113	80	.50	1982-03-01	1982-04-15
	000260	AD3113	180	.50	1982-03-01	1982-04-15
	000260	AD3113	180	1.00	1982-04-15	1982-06-01
	000260	AD3113	180	.50	1982-06-01	1982-07-01
	000270	AD3113	60	.50	1982-03-01	1982-04-01
	000270	AD3113	60	1.00	1982-04-01	1982-09-01
	000270	AD3113	60	.25	1982-09-01	1982-10-15
	000270	AD3113	70	.75	1982-09-01	1982-10-15
	000270	AD3113	70	1.00	1982-10-15	1983-02-01
	000270	AD3113	80	1.00	1982-01-01	1982-03-01

이름:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000270	AD3113	80	.50	1982-03-01	1982-04-01
	000030	IF1000	10	.50	1982-06-01	1983-01-01
	000130	IF1000	90	1.00	1982-01-01	1982-10-01
	000130	IF1000	100	.50	1982-10-01	1983-01-01
	000140	IF1000	90	.50	1982-10-01	1983-01-01
	000030	IF2000	10	.50	1982-01-01	1983-01-01
	000140	IF2000	100	1.00	1982-01-01	1982-03-01
	000140	IF2000	100	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-10-01	1983-01-01
	000010	MA2100	10	.50	1982-01-01	1982-11-01
	000110	MA2100	20	1.00	1982-01-01	1982-03-01
	000010	MA2110	10	1.00	1982-01-01	1983-02-01
	000200	MA2111	50	1.00	1982-01-01	1982-06-15
	000200	MA2111	60	1.00	1982-06-15	1983-02-01
	000220	MA2111	40	1.00	1982-01-01	1983-02-01
	000150	MA2112	60	1.00	1982-01-01	1982-07-15
	000150	MA2112	180	1.00	1982-07-15	1983-02-01
	000170	MA2112	60	1.00	1982-01-01	1983-06-01
	000170	MA2112	70	1.00	1982-06-01	1983-02-01
	000190	MA2112	70	1.00	1982-02-01	1982-10-01
	000190	MA2112	80	1.00	1982-10-01	1983-10-01
	000160	MA2113	60	1.00	1982-07-15	1983-02-01
	000170	MA2113	80	1.00	1982-01-01	1983-02-01
	000180	MA2113	70	1.00	1982-04-01	1982-06-15
	000210	MA2113	80	.50	1982-10-01	1983-02-01
	000210	MA2113	180	.50	1982-10-01	1983-02-01
	000050	OP1000	10	.25	1982-01-01	1983-02-01
	000090	OP1010	10	1.00	1982-01-01	1983-02-01
	000280	OP1010	130	1.00	1982-01-01	1983-02-01
	000290	OP1010	130	1.00	1982-01-01	1983-02-01
	000300	OP1010	130	1.00	1982-01-01	1983-02-01
	000310	OP1010	130	1.00	1982-01-01	1983-02-01
	000050	OP2010	10	.75	1982-01-01	1983-02-01
	000100	OP2010	10	1.00	1982-01-01	1983-02-01
	000320	OP2011	140	.75	1982-01-01	1983-02-01
	000320	OP2011	150	.25	1982-01-01	1983-02-01

이름:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000330	OP2012	140	.25	1982-01-01	1983-02-01
	000330	OP2012	160	.75	1982-01-01	1983-02-01
	000340	OP2013	140	.50	1982-01-01	1983-02-01
	000340	OP2013	170	.50	1982-01-01	1983-02-01
	000020	PL2100	30	1.00	1982-01-01	1982-09-15

EMP_PHOTO 테이블

이름:	EMPNO	PHOTO_FORMAT	PICTURE
유형:	char(6) 널(NULL)이 아님	varchar(10) 널(NULL)이 아님	blob(100k)
Desc:	사원 번호	사진 양식	사원 사진
값:	000130	비트맵	db200130.bmp
	000130	gif	db200130.gif
	000130	xwd	db200130.xwd
	000140	비트맵	db200140.bmp
	000140	gif	db200140.gif
	000140	xwd	db200140.xwd
	000150	비트맵	db200150.bmp
	000150	gif	db200150.gif
	000150	xwd	db200150.xwd
	000190	비트맵	db200190.bmp
	000190	gif	db200190.gif
	000190	xwd	db200190.xwd

- 1434 페이지의 『Quintana 사진』에서는 Delores Quintana 사원의 사진을 보여줍니다.
- 1436 페이지의 『Nicholls 사진』에서는 Heather Nicholls 사원의 사진을 보여줍니다.
- 1438 페이지의 『Adamson 사진』에서는 Bruce Adamson 사원의 사진을 보여줍니다.
- 1440 페이지의 『Walker 사진』에서는 James Walker 사원의 사진을 보여줍니다.

EMP_RESUME 테이블

이름:	EMPNO	RESUME_FORMAT	RESUME
유형:	char(6) 널(NULL)이 아님	varchar(10) 널(NULL)이 아님	clob(5k)
Desc:	사원 번호	이력서 양식	사원의 이력서
값:	000130	ASCII	db200130.asc
	000130	스크립트	db200130.scr
	000140	ASCII	db200140.asc
	000140	스크립트	db200140.scr
	000150	ASCII	db200150.asc
	000150	스크립트	db200150.scr
	000190	ASCII	db200190.asc
	000190	스크립트	db200190.scr

- 1434 페이지의 『Quintana 이력서』에서는 Delores Quintana 사원의 이력서를 보여줍니다.
- 1436 페이지의 『Nicholls 이력서』에서는 Heather Nicholls 사원의 이력서를 보여줍니다.
- 1438 페이지의 『Adamson 이력서』에서는 Bruce Adamson 사원의 이력서를 보여줍니다.
- 1440 페이지의 『Walker 이력서』에서는 James Walker 사원의 이력서를 보여줍니다.

IN_TRAY 테이블

이름:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
유형:	시간소인	char(8)	char(64)	varchar(3000)
Desc:	접수된 날짜 및 시간	주를 송신한 사람의 사용자 ID	간단한 설명	주

ORG 테이블

이름:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
유형:	smallint 널(NULL)이 아님	varchar(14)	smallint	varchar(10)	varchar(13)
Desc:	부서 번호	부서 이름	관리자 번호	회사의 지사	도시

이름:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
값:	10	Head Office	160	본사	뉴욕
	15	뉴잉글랜드	50	동부	보스톤
	20	중부 아틀란틱	10	동부	워싱톤
	38	남부 아틀란틱	30	동부	아틀란타
	42	Great Lakes	100	중서부	시카고
	51	Plains	140	중서부	달라스
	66	Pacific	270	서부	샌프란시스코
	84	Mountain	290	서부	덴버

PROJECT 테이블

이름:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
유형:	char(6) 널(NULL)이 아님	varchar(24) 널(NULL)이 아님	char(3) 널(NULL)이 아님	char(6) 널(NULL)이 아님	dec(5,2)	날짜	날짜	char(6)
Desc:	프로젝트 번호	프로젝트 이름	부서 책임분	사원 책임분	예상 평균 인원	예상 시작 날짜	예상 종료 날짜	서브프로젝트에 대한 주요 프로젝트
값:	AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	-
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
	IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	-
	IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	-
	MA2100	WELD LINE AUTOMATION	D01	000010	12	1982-01-01	1983-02-01	-
	MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
	MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110

이름:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
	MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
	OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	-
	OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	-
	OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

SALES 테이블

이름:	SALES_DATE	SALES_PERSON	REGION	SALES
유형:	날짜 시간	varchar(15)	varchar(15)	int
Desc:	봉급 날짜	사원의 성	영업 지역	영업 번호
값:	12/31/1995	LUCCHESSI	Ontario-South	1
	12/31/1995	LEE	Ontario-South	3
	12/31/1995	LEE	Quebec	1
	12/31/1995	LEE	Manitoba	2
	12/31/1995	GOUNOT	Quebec	1
	03/29/1996	LUCCHESSI	Ontario-South	3
	03/29/1996	LUCCHESSI	Quebec	1
	03/29/1996	LEE	Ontario-South	2
	03/29/1996	LEE	Ontario-North	2
	03/29/1996	LEE	Quebec	3
	03/29/1996	LEE	Manitoba	5
	03/29/1996	GOUNOT	Ontario-South	3
	03/29/1996	GOUNOT	Quebec	1
	03/29/1996	GOUNOT	Manitoba	7
	03/30/1996	LUCCHESSI	Ontario-South	1
	03/30/1996	LUCCHESSI	Quebec	2
	03/30/1996	LUCCHESSI	Manitoba	1

이름:	SALES_DATE	SALES_PERSON	REGION	SALES
	03/30/1996	LEE	Ontario-South	7
	03/30/1996	LEE	Ontario-North	3
	03/30/1996	LEE	Quebec	7
	03/30/1996	LEE	Manitoba	4
	03/30/1996	GOUNOT	Ontario-South	2
	03/30/1996	GOUNOT	Quebec	18
	03/30/1996	GOUNOT	Manitoba	1
	03/31/1996	LUCCHESSI	Manitoba	1
	03/31/1996	LEE	Ontario-South	14
	03/31/1996	LEE	Ontario-North	3
	03/31/1996	LEE	Quebec	7
	03/31/1996	LEE	Manitoba	3
	03/31/1996	GOUNOT	Ontario-South	2
	03/31/1996	GOUNOT	Quebec	1
	04/01/1996	LUCCHESSI	Ontario-South	3
	04/01/1996	LUCCHESSI	Manitoba	1
	04/01/1996	LEE	Ontario-South	8
	04/01/1996	LEE	Ontario-North	-
	04/01/1996	LEE	Quebec	8
	04/01/1996	LEE	Manitoba	9
	04/01/1996	GOUNOT	Ontario-South	3
	04/01/1996	GOUNOT	Ontario-North	1
	04/01/1996	GOUNOT	Quebec	3
	04/01/1996	GOUNOT	Manitoba	7

STAFF 테이블

이름:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
유형:	smallint 널(NULL)이 아님	varchar(9)	smallint	char(5)	smallint	dec(7,2)	dec(7,2)
Desc:	사원 번호	사원 이름	부서 번호	직위	근무 년수	현재 봉급	커미션
값:	10	Sanders	20	Mgr	7	18357.50	-
	20	개인	20	영업	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	영업	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	영업	-	16808.30	650.25
	70	Rothman	15	영업	7	16502.83	1152.00
	80	James	20	사원	-	13504.60	128.20

이름:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
	90	Koonitz	42	영업	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	사원	5	12508.20	206.60
	120	Naughton	38	사원	-	12954.75	180.00
	130	Yamaguchi	42	사원	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	영업	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	사원	4	12258.50	110. 10
	180	Abrahams	38	사원	3	12009. 75	236. 50
	190	Sneider	20	사원	8	14252. 75	126. 50
	200	Scoutten	42	사원	-	11508. 60	84. 20
	210	Lu	10	Mgr	10	20010. 00	-
	220	Smith	51	영업	7	17654. 50	992. 80
	230	Lundquist	51	사원	3	13369. 80	189. 65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	사원	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	영업	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	영업	5	15454.50	806.10
	310	Graham	66	영업	13	21000.00	200.30
	320	Gonzales	66	영업	4	16858.20	844.00
	330	Burke	66	사원	1	10988.00	55.50
	340	Edwards	84	영업	7	17844.00	1285.00
	350	Gafney	84	사원	5	13030.50	188.00

STAFFG 테이블

주: STAFFG는 더블 바이트 코드 페이지에 대해서만 생성됩니다.

이름:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
유형:	smallint 널(NULL)이 아님	vargraphic(9)	smallint	graphic(5)	smallint	dec(9,0)	dec(9,0)
Desc:	사원 번호	사원 이름	부서 번호	직위	근무 년수	현재 봉급	커미션

이름:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
값:	10	Sanders	20	Mgr	7	18357.50	-
	20	개인	20	영업	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	영업	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	영업	-	16808.30	650.25
	70	Rothman	15	영업	7	16502.83	1152.00
	80	James	20	사원	-	13504.60	128.20
	90	Koonitz	42	영업	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	사원	5	12508.20	206.60
	120	Naughton	38	사원	-	12954.75	180.00
	130	Yamaguchi	42	사원	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	영업	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	사원	4	12258.50	110. 10
	180	Abrahams	38	사원	3	12009. 75	236. 50
	190	Sneider	20	사원	8	14252. 75	126. 50
	200	Scoutten	42	사원	-	11508. 60	84. 20
	210	Lu	10	Mgr	10	20010. 00	-
	220	Smith	51	영업	7	17654. 50	992. 80
	230	Lundquist	51	사원	3	13369. 80	189. 65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	사원	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	영업	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	영업	5	15454.50	806.10
	310	Graham	66	영업	13	21000.00	200.30
	320	Gonzales	66	영업	4	16858.20	844.00
	330	Burke	66	사원	1	10988.00	55.50
	340	Edwards	84	영업	7	17844.00	1285.00
	350	Gafney	84	사원	5	13030.50	188.00

BLOB 및 CLOB 데이터 유형으로 된 샘플 파일

이 절에서는 EMP_PHOTO 파일(직원 사진)과 EMP_RESUME 파일(직원 이력서)에 있는 데이터를 보여줍니다.

Quintana 사진

그림 13. Delores M. Quintana

Quintana 이력서

다음과 같은 텍스트는 db200130.asc 및 db200130.scr 파일에 있습니다.

이력서: Delores M. Quintana

개인 정보	
주소:	1150 Eglinton Ave Mellonville, Idaho 83725
전화번호:	(208) 555-9933
생년월일:	September 15, 1925
성:	Female
결혼 여부:	Married
키:	5’2″
체중:	120 lbs.

부서 정보

사원 번호: 000130
부서 번호: C01
관리자: Sally Kwan
직위: 분석자
전화번호: (208) 555-4578
입사일: 1971-07-28

학력

1965 Math and English, B.A. Adelphi University

1960 Dental Technician Florida Institute of Technology

작업 실행기록

10/91 - 현재 Advisory Systems Analyst Producing documentation tools for engineering department.

12/85 - 9/91 Technical Writer Writer, text programmer, and planner.

1/79 - 11/85 COBOL Payroll Programmer Writing payroll programs for a diesel fuel company.

특기사항

- Cooking
- Reading
- Sewing
- Remodeling

Nicholls 사진

그림 14. Heather A. Nicholls

Nicholls 이력서

다음과 같은 텍스트는 db200140.asc 및 db200140.scr 파일에 있습니다.

이력서: Heather A. Nicholls

개인 정보	
주소:	844 Don Mills Ave Mellonville, Idaho 83734
전화번호:	(208) 555-2310
생년월일:	January 19, 1946
성별:	Female
결혼 여부:	Single
키:	5’8″
체중:	130 lbs.
부서 정보	
사원 번호:	000140
부서 번호:	C01
관리자:	Sally Kwan

직위:	Analyst
전화번호:	(208) 555-1793
입사일:	1976-12-15

학력

1972	Computer Engineering, Ph.D. University of Washington
1969	Music and Physics, M.A. Vassar College

작업 실행기록

2/83 - 현재	Architect, OCR Development Designing the architecture of OCR products.
12/76 - 1/83	Text Programmer Optical character recognition (OCR) programming in PL/I.
9/72 - 11/76	Punch Card Quality Analyst Checking punch cards met quality specifications.

특기사항

- Model railroading
- Interior decorating
- Embroidery
- Knitting

Adamson 사진

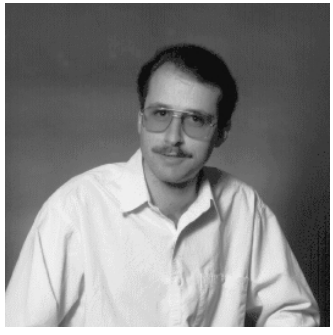


그림 15. Bruce Adamson

Adamson 이력서

다음과 같은 텍스트는 db200150.asc 및 db200150.scr 파일에 있습니다.

이력서: Bruce Adamson

개인 정보	
주소:	3600 Steeles Ave Mellonville, Idaho 83757
전화번호:	(208) 555-4489
생년월일:	May 17, 1947
성별:	Male
결혼 여부:	Married
키:	6’0″
체중:	175 lbs.
부서 정보	
사원 번호:	000150
부서 번호:	D11
관리자:	Irving Stern

직위:	Designer
전화번호:	(208) 555-4510
입사일:	1972-02-12

학력

1971	Environmental Engineering, M.Sc. Johns Hopkins University
1968	American History, B.A. Northwestern University

작업 실행기록

8/79 - 현재	Neural Network Design Developing neural networks for machine intelligence products.
2/72 - 7/79	Robot Vision Development Developing rule-based systems to emulate sight.
9/71 - 1/72	Numerical Integration Specialist Helping bank systems communicate with each other.

특기사항

- Racing motorcycles
- Building loudspeakers
- Assembling personal computers
- Sketching

Walker 사진

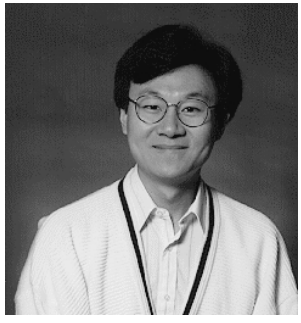


그림 16. James H. Walker

Walker 이력서

다음과 같은 텍스트는 db200190.asc 및 db200190.scr 파일에 있습니다.

이력서: James H. Walker

개인 정보	
주소:	3500 Steeles Ave Mellonville, Idaho 83757
전화번호:	(208) 555-7325
생년월일:	June 25, 1952
성별:	Male
결혼 여부:	Single
키:	5’11″
체중:	166 lbs.
부서 정보	
사원 번호:	000190
부서 번호:	D11
관리자:	Irving Stern
직위:	Designer

전화번호: (208) 555-2986
입사일: 1974-07-26

학력

1974 Computer Studies, B.Sc. University of Massachusetts

1972 Linguistic Anthropology, B.A. University of Toronto

작업 실행기록

6/87 - 현재 Microcode Design Optimizing algorithms for mathematical functions.

4/77 - 5/87 Printer Technical Support Installing and supporting laser printers.

9/74 - 3/77 Maintenance Programming Patching assembly language compiler for mainframes.

특기사항

- Wine tasting
- Skiing
- Swimming
- Dancing

부록H. 예약된 스키마 이름 및 예약어

이 부록은 데이터베이스 관리 프로그램이 사용하는 일부 이름의 제한사항을 기술합니다. 이름이 예약되어 있어 응용프로그램에서 사용할 수 없는 경우도 있습니다. 어떤 경우에는 데이터베이스 관리 프로그램에서 제약을 두는 것은 아니지만, 응용프로그램에서 사용하지 않는 것이 바람직한 이름도 있습니다.

예약된 스키마

다음 스키마 이름이 예약되어 있습니다.

- SYSCAT
- SYSFUN
- SYSIBM
- SYSSTAT

또한, SYS는 규칙에 의해 시스템에서 예약된 영역을 표시하는 데 사용되므로, 스키마 이름은 SYS 접두어로 시작해서는 안 됩니다.

사용자 정의 함수, 사용자 정의 유형, 트리거 또는 별명을 SYS로 시작하는 이름의 스키마에 둘 수 없습니다(SQLSTATE 42939).

스키마 이름으로 SESSION을 사용하지 말 것을 권장합니다. 선언된 임시 테이블은 SESSION으로 규정되어야 하므로, 응용프로그램에서 영속 테이블의 이름과 동일한 이름을 갖는 임시 테이블을 선언할 수 있는데 이는 응용프로그램 로직을 복잡하게 만듭니다. 이러한 가능성을 없애기 위해서는 선언된 임시 테이블을 처리할 때를 제외하고는 스키마 SESSION을 사용하지 마십시오.

예약어

DB2 버전 7에 특별히 예약된 단어는 없습니다.

키워드는 SQL 키워드로도 해석될 수 있는 문맥을 제외하고는 일반적인 식별자로 사용될 수 있습니다. 그러한 경우, 단어는 분리 식별자로 지정되어야 합니다. 예를 들어, COUNT는 분리되지 않는 한 SELECT문에서 컬럼 이름으로 사용될 수 없습니다.

IBM SQL 및 ISO/ANSI SQL92에는 다음 섹션에 나열되어 있는 예약된 단어가 포함되어 있습니다. 이들 예약어는 DB2 Universal Database에서 사용되지 않습니다. 그러나, 이들은 이식성을 저하시키므로 일반 식별자로 사용하지 않는 것이 좋습니다.

IBM SQL 예약어

IBM SQL 예약어는 다음과 같습니다.

ACQUIRE
ADD
AFTER
ALIAS
ALL
ALLOCATE
ALLOW
ALTER
AND
ANY
AS
ASC
ASUTIME
AUDIT
AUTHORIZATION
AUX
AUXILIARY
AVG

BEFORE
BEGIN
BETWEEN
BINARY
BUFFERPOOL
BY

CALL
CALLED
CAPTURE
CASCADED
CASE
유형변환(CAST)
CCSID
CHAR
CHARACTER
CHECK
CLOSE
CLUSTER
COLLECTION
COLLID
COLUMN
COMMENT
COMMIT
CONCAT
CONDITION
CONNECT
CONNECTION
CONSTRAINT
CONTAINS
CONTINUE
COUNT
COUNT_BIG
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_LC_PATH
CURRENT_PATH
CURRENT_SERVER
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_TIMEZONE
CURRENT_USER
CURSOR

DATA
DATABASE
DATE
DAY
DAYS
DBA
DBINFO
DBSPACE
DB2GENERAL
DB2SQL
DECLARE
DEFAULT
DELETE
DESC
DESCRIPTOR
DETERMINISTIC
DISALLOW
DISCONNECT
DISTINCT
DO
DOUBLE
DROP
DSSIZE
DYNAMIC
EDITPROC
ELSE
ELSEIF
END
END-EXEC
ERASE
ESCAPE
EXCEPT
EXCEPTION
EXCLUSIVE
EXECUTE
EXISTS
EXIT
EXPLAIN
EXTERNAL

FENCED
FETCH
FIELDPROC
FILE
FINAL
FOR
FOREIGN
FREE
FROM
FULL
FUNCTION

GENERAL
GENERATED
GO
GOTO
GRANT
GRAPHIC
GROUP

HANDLER
HAVING
HOUR
HOURS

IDENTIFIED
IF
IMMEDIATE
IN
INDEX
INDICATOR
INNER
INOUT
INSENSITIVE
INSERT
INTEGRITY
INTERSECT
INTO
IS
ISOBID
ISOLATION

JAVA
JOIN

KEY

LABEL
LANGUAGE
LC_CTYPE
LEAVE
LEFT
LIKE
LINKTYPE
LOCAL
LOCALE
LOCATOR
LOCATORS
LOCK
LOCKSIZE
LONG
LOOP

MAX
MICROSECOND
MICROSECONDS
MIN
MINUTE
MINUTES
MODE
MODIFIES
MONTH
MONTHS

NAME
NAMED
NHEADER
NO
NODENAME
NODENUMBER
NOT
NULL
NULLS
NUMPARTS

OBID
OF
ON
ONLY
OPEN
OPTIMIZATION
OPTIMIZE
OPTION
또는
ORDER
OUT
OUTER

PACKAGE
PAGE
PAGES
PARAMETER
PART
PARTITION
PATH
PCTFREE
PCTINDEX
PIECESIZE
PLAN
POSITION
PRECISION
PREPARE
PRIMARY
PRIQTY
PRIVATE
PRIVILEGES
PROCEDURE
PROGRAM
PSID
PUBLIC

QUERYNO

READ
READS
RECOVERY
REFERENCES
RELEASE
RENAME
REPEAT
RESET
RESOURCE
RESTRICT
RESULT
RETURN
RETURNS
REVOKE
RIGHT
ROLLBACK
ROW
ROWS
RRN
RUN

SCHEDULE
SCHEMA
SCRATCHPAD
SECOND
SECONDS
SECQTY
SECURITY
SELECT
SET
SHARE
SIMPLE
SOME
SOURCE
SPECIFIC
SQL
STANDARD
STATIC
STATISTICS
STAY
STOGROUP
STORES
STORPOOL
STYLE
SUBPAGES
SUBSTRING
SUM
SYNONYM

TABLE
TABLESPACE
THEN
TO
TRANSACTION
TRIGGER
TRIM
TYPE

UNDO
UNION
UNIQUE
UNTIL
UPDATE
USAGE
USER
USING

VALIDPROC
VALUES
VARIABLE
VARIANT
VCAT
VIEW
VOLUMES

WHEN
WHERE
WHILE
WITH
WLM
WORK
WRITE

YEAR
YEARS

ISO/ANS SQL92 예약어

IBM SQL 목록에 들어 있지 않은 ISO/ANS SQL92 예약어는 다음과 같습니다.

ABSOLUTE
ACTION
ARE
ASSERTION
AT

BIT_LENGTH
BOTH

CATALOG
CHAR_LENGTH
CHARACTER_LENGTH
COALESCE
COLLATE
COLLATION
CONSTRAINTS
CONVERT
CORRESPONDING

DEALLOCATE
DEC
DECIMAL
DEFERRABLE
DEFERRED
DESCRIBE
DIAGNOSTICS
DOMAIN

EXEC
EXTRACT

FALSE
FIRST
FLOAT
FOUND
FULL

GET
GLOBAL

IDENTITY
INITIALLY
INPUT
INTERVAL

LAST
LEADING
LEVEL
LOWER

MATCH
MODULE

NAMES
NATIONAL
NATURAL
NCHAR
NEXT
NULLIF
NUMERIC

OCTET_LENGTH
OUTPUT
OVERLAPS

PAD
PARTIAL
PRESERVE
PRIOR

REAL
RELATIVE

SCROLL
SECTION
SESSION
SESSION_USER
SIZE
SMALLINT
SPACE
SQLCODE
SQLERROR
SQLSTATE
SYSTEM_USER

TEMPORARY
TIMEZONE_HOUR
TIMEZONE_MINUTE
TRAILING
TRANSLATION
TRUE

UNKNOWN
UPPER

VALUE
VARCHAR
VARYING

WHENEVER

ZONE

부록I. 분리 레벨 비교

다음 표는 30 페이지의 『분리 레벨』에 기술된 분리 레벨에 관한 정보를 요약한 것입니다.

	UR	CS	RS	RR
응용프로그램이 다른 응용프로그램 프로세스에 의해 변경된 비확약 변경사항을 볼 수 있습니까?	예	아니오	아니오	아니오
응용프로그램이 다른 응용프로그램 프로세스에 의해 변경된 비확약 변경사항을 갱신할 수 있습니까?	아니오	아니오	아니오	아니오
명령문이 재실행될 때 다른 응용프로그램 프로세스에 의해 영향 받습니까? *아래의 P3(가상) 현상 참조.*	예	예	예	아니오
"갱신된" 행이 다른 응용프로그램 프로세스에 의해 갱신될 수 있습니까?	아니오	아니오	아니오	아니오
UR이 아닌 분리 레벨에서 수행 중인 다른 응용프로그램 프로세스가 "갱신된" 행을 읽을 수 있습니까?	아니오	아니오	아니오	아니오
UR 분리 레벨에서 수행 중인 다른 응용프로그램 프로세스가 "갱신된" 행을 읽을 수 있습니까?	예	예	예	예
다른 응용프로그램프로세스에 의해 "액세스된" 행이 갱신될 수 있습니까? *아래의 P2(반복 불가능한 읽기) 참조.*	예	예	아니오	아니오
다른 응용프로그램 프로세스가 "액세스된" 행을 읽을 수 있습니까?	예	예	예	예
다른 응용프로그램 프로세스가 "현재의" 행을 갱신하거나 삭제할 수 있습니까? *아래의 P1(더티-읽기) 참조.*	주 참조	주 참조	아니오	아니오

주:

1. 커서가 갱신 가능하지 않은 경우, 어떤 경우에는 CS로 현재 행을 다른 응용프로그램 프로세스가 갱신 또는 삭제하기도 합니다.

	UR	CS	RS	RR

현상의 예:

P1 *더티 읽기.* 작업 단위(UOW) UW1에서 행을 수정합니다. 작업 단위(UOW) UW2는 UW1에서 COMMIT을 수행하기 전에 행을 읽습니다. 그런 후 UW1이 ROLLBACK을 수행하면, UW2는 존재하지 않는 행을 읽은 것이 됩니다.

P2 *반복 불가능한 읽기.* 작업 단위(UOW) UW1에서 행을 읽습니다. 작업 단위(UOW) UW2가 행을 수정하고 COMMIT을 수행합니다. 그런 후 UW1이 행을 다시 읽으면, 수정된 값을 받을 수도 있습니다.

P3 *팬텀.* 작업 단위(UOW) UW1이 일부 검색 조건을 만족시키는 *n* 행 집합을 읽습니다. 그런 후 작업 단위 UW2는 검색 조건을 만족시키는 하나 이상의 행을 INSERT합니다. UW1이 같은 검색 조건으로 첫번째 읽기를 반복하면, 원래의 행과 삽입된 행을 얻습니다.

부록J. 트리거와 제한조건의 상호 작용

이 부록에서는 갱신으로 발생할 수 있는 참조 제한조건 및 점검 제한조건과 트리거와의 상호 작용을 설명합니다. 그림17 및 관련 설명은 데이터베이스 내의 데이터를 갱신하는 SQL문에 대해 수행된 처리의 결과입니다.

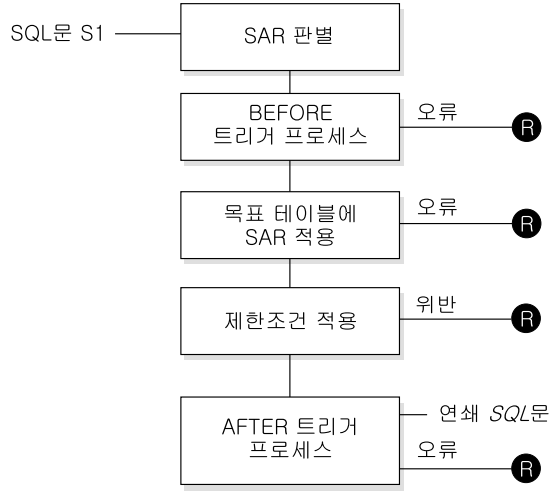


그림 17. 관련 트리거 및 제한조건으로 SQL문 처리

그림17은 테이블을 갱신하는 SQL문의 일반적인 처리 순서를 보여줍니다. 테이블에는 사전(before) 트리거, 참조 제한조건, 점검 제한조건 및 사후 트리거가 연쇄적으로 들어 있다고 가정합니다. 다음의 위 그림과 그림17에 있는 그외 항목에 대한 설명입니다.

- SQL문 S_1

  프로세스에서 시작하는 DELETE, INSERT 또는 UPDATE문입니다. SQL문 S_1은 이 설명에서 *목표 테이블*로 지칭되는 테이블(또는 다른 일부 테이블에 대한 갱신 가능 뷰)을 가리킵니다.

- 영향받는 행의 세트 판별(*SAR*)

이 단계는 CASCADE 및 SET NULL의 참조 제한조건 삭제 규칙과 사후 트리거로부터의 연쇄 SQL문을 반복하는 프로세스의 시작점입니다.

이 단계의 목적은 SQL문을 *만족시키는 행의 세트*를 판별하는 것입니다. *SAR*에 포함된 행 세트는 다음 명령문을 근거로 합니다.

- DELETE의 경우, 명령문의 검색 조건을 만족하는 모든 행(또는, 위치 지정된 DELETE에 대한 현재 행)
- INSERT의 경우, VALUES절 또는 fullselect가 식별하는 행
- UPDATE의 경우, 검색 조건을 만족하는 모든 행(또는 위치 지정된 UPDATE에 대한 현재 행)

*SAR*이 비어 있는 경우에는, BEFORE 트리거, 목표 테이블에 적용된 변경사항 또는 SQL문을 처리하기 위한 의무 규정 등이 없습니다.

- BEFORE 트리거 프로세스

  모든 BEFORE 트리거는 작성의 오름차순으로 처리됩니다. 각 BEFORE 트리거는 *SAR*에 있는 각 행에 대해 한번의 트리거 조치를 처리합니다.

  트리거 조치 처리중 오류가 발생할 수 있으며, 이 경우 원본 SQL문 S_1의 결과 만들어진 모든 변경은 구간 복원됩니다.

  BEFORE 트리거가 없거나 *SAR*이 비어 있는 경우, 이 단계는 생략됩니다.

- 목표 테이블에 *SAR* 적용

  실제 삭제, 삽입 또는 갱신은 *SAR*을 사용하여 데이터베이스 내의 목표 테이블에 적용됩니다.

  SAR(고유 색인이 존재하는 중복 키에 행을 삽입하려는 것 등)을 적용할 경우 오류가 발생할 수 있으며, 이 경우 원본 SQL문 S_1의 결과 만들어진 모든 변경은 구간 복원됩니다.

- 제한조건 적용

  목표 테이블과 관련된 제한조건은 *SAR*이 빈 경우 적용됩니다. 여기에는 고유 제한조건, 고유 색인, 참조 제한조건, 점검 제한조건 및 뷰에 대한 WITH CHECK OPTION과 관련한 점검이 포함됩니다. 연쇄 또는 널(NULL) 설정의 삭제 규칙을 가진 참조 제한조건은 추가의 트리거를 활성화시킵니다.

  제한조건이나 WITH CHECK OPTION 위반시 오류가 초래되며 S_1의 결과 발생된 모든 변경이 구간 복원됩니다.

- AFTER 트리거 프로세스

S_1에 의해 활성화된 모든 AFTER 트리거는 오름차순으로 처리됩니다.

FOR EACH STATEMENT 트리거는 *SAR*이 빈 경우, 단 한번의 트리거 조치를 처리합니다. FOR EACH ROW 트리거는 *SAR*에 있는 각 행에 대해 한번의 트리거 조치를 처리합니다.

트리거 조치 처리중 오류가 발생할 수 있으며, 이 경우 원본 S_1의 결과 만들어진 모든 변경은 구간 복원됩니다.

트리거 SQL문을 포함하는 트리거의 트리거 조치에는 DELETE, INSERT 또는 UPDATE문이 포함됩니다. 이 설명의 목적을 위해 각 명령문은 *연쇄 SQL*문으로 간주됩니다.

연쇄 SQL문은 AFTER 트리거의 트리거 조치 일부로 처리되는 DELETE, INSERT 또는 UPDATE문입니다. 이 명령문은 트리거 처리의 연쇄 레벨로 시작합니다. 이것은 트리거 SQL문을 새로운 S_1로 간주하고 여기에 기술된 모든 단계를 반복적으로 수행하는 것으로 생각할 수 있습니다.

일단 S_1에 의해 활성화된 모든 AFTER 트리거의 모든 SQL문이 완전히 처리되면, 원본 S_1의 처리가 완료됩니다.

- **R** = 변경을 구간 복원 S_1

  처리중 발생하는 오류(제한조건 위반 등)는 원본 SQL문 S_1의 결과 직접 또는 간접적으로 만들어진 모든 변경을 구간 복원합니다. 그러므로, 데이터베이스는 원본 SQL문 S_1의 실행 바로 전과 같은 상태로 돌아갑니다.

부록K. Explain 테이블 및 정의

설명 기능이 활성화될 때 Explain 테이블 캡쳐 액세스 플랜합니다. 이 절에서 설명된 Explain 테이블 및 정의는 다음과 같습니다.

- 1456 페이지의 『EXPLAIN_ARGUMENT 테이블』
- 1460 페이지의 『EXPLAIN_INSTANCE 테이블』
- 1463 페이지의 『EXPLAIN_OBJECT 테이블』
- 1465 페이지의 『EXPLAIN_OPERATOR 테이블』
- 1467 페이지의 『EXPLAIN_PREDICATE 테이블』
- 1469 페이지의 『EXPLAIN_STATEMENT 테이블』
- 1472 페이지의 『EXPLAIN_STREAM 테이블』
- 1473 페이지의 『ADVISE_INDEX 테이블』
- 1476 페이지의 『ADVISE_WORKLOAD 테이블』

Explain을 호출하려면 Explain 테이블을 먼저 작성해야 합니다. 테이블을 작성하려면, 'sqllib' 디렉토리의 'misc' 서브디렉토리에 있는 EXPLAIN.DDL 파일에 제공된 샘플 명령행 프로세서 입력 스크립트를 사용하십시오. Explain 테이블이 필요한 데이터베이스와 연결하십시오. 그리고 나서 작성될 명령: `db2 -tf EXPLAIN.DDL` 및 테이블을 발행하십시오. 1477 페이지의 『Explain 테이블에 대한 테이블 정의』에서 자세한 정보를 참조하십시오.

설명 기능으로 모아진 Explain 테이블의 모임은 트리거를 활성화하지 못하거나 참조나 점검 제한조건도 활성화하지 못합니다. 예를 들어, 삽입 트리거를 EXPLAIN_INSTANCE 테이블에 정의하고 적당한 명령문을 설명할 경우, 트리거는 활성화되지 못합니다.

설명 기능에 대한 *관리 안내서*에서 자세한 내용을 참조하십시오.

Explain 테이블에 대한 설명:

헤드 **Explain**

컬럼 이름	컬럼 이름
데이터 유형	컬럼의 데이터 유형.
널(NULL) 입력 가능?	예: 널(NULL) 허용 아니오: 널(NULL) 허용 안됨
키?	PK: 컬럼이 기본 키의 일부 FK: 컬럼이 외부 키의 일부
설명	컬럼의 설명

EXPLAIN_ARGUMENT 테이블

모든 경우에, EXPLAIN_ARGUMENT 테이블은 개개의 각 연산자에 대해 고유한 특성을 나타냅니다.

이 테이블의 정의에 대해서는 1478 페이지의 『EXPLAIN_ARGUMENT 테이블 정의』에서 참조하십시오.

표 128. EXPLAIN_ARGUMENT 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	이 Explain 요청의 개시자 권한 부여 ID.
EXPLAIN_TIME	TIMESTAMP	아니오	FK	Explain 요청에 대한 시작 시간.
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적 명령문이 설명될 때 수행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름.
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청 소스의 스키마, 규정자.
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행과 관련한 Explain 정보의 레벨.
STMTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호.
OPERATOR_ID	INTEGER	아니오	아니오	이 조회 내의 연산자에 대한 고유 ID.
ARGUMENT_TYPE	CHAR(8)	아니오	아니오	이 연산자에 대한 인수의 유형.
ARGUMENT_VALUE	VARCHAR(1024)	예	아니오	이 연산자에 대한 인수의 값. 값이 LONG_ARGUMENT_VALUE에 있는 경우에는 널(NULL).
LONG_ARGUMENT_VALUE	CLOB(1M)	예	아니오	텍스트가 ARGUMENT_VALUE에 맞지 않을 때에는 이 연산자에 대한 인수의 값. 값이 ARGUMENT_VALUE에 있는 경우에는 널(NULL).

표 129. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	부분 총계 표시기
BITFLTR	TRUE FALSE	해쉬 조인은 비트 필터를 사용하여 성능을 향상시킵니다.
CSETEMP	TRUE FALSE	일반 부속 표현식 플래그에 대한 일시 테이블.
DIRECT	TRUE	직접 패치 표시기
DUPLWARN	TRUE FALSE	경고 플래그를 중복합니다.
EARLYOUT	TRUE FALSE	Early out 표시기.
ENVVAR	이 유형의 각 행은 다음을 포함하고 있습니다. • 환경 변수 이름 • 환경 변수 값	최적화 알고리즘에 영향을 미치는 환경 변수
FETCHMAX	IGNORE INTEGER	FETCH 연산자에서 MAXPAGES 인수의 겹쳐쓰기 값.
GROUPBYC	TRUE FALSE	Group By 컬럼이 제공된 경우.
GROUPBYN	정수	비교 컬럼의 수.
GROUPBYR	이 유형의 각 행은 다음을 포함하고 있습니다. • 절별 그룹에 있는 컬럼의 순서 값 (이 뒤에는 콜론과 공간이 옴) • 컬럼 이름	Group By 요구.
INNERCOL	이 유형의 각 행은 다음을 포함하고 있습니다. • 순서대로 된 순서 값 (콜론과 공간이 뒤에 따라오는) • 컬럼 이름 • 순서 값 **(A)** 오름차순 **(D)** 내림차순	내부 순서 컬럼
ISCANMAX	IGNORE INTEGER	ISCAN 연산자의 MAXPAGES 인수에 대한 겹쳐쓰기 값.
JN_INPUT	INNER OUTER	연산자가 조인의 내부 및 외부를 산출하는지를 나타냅니다.

표 129. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
LISTENER	TRUE FALSE	대기자 테이블 대기행렬 표시기.
MAXPAGES	ALL NONE INTEGER	프리패치에 대해 기대되는 최대 페이지
MAXRIDS	NONE INTEGER	각 목록 프리페치 요청에 포함될 최대 행 식별자.
NUMROWS	INTEGER	저장되려는 행의 수
ONEFETCH	TRUE FALSE	한개 패치 표시기
OUTERCOL	이 유형의 각 행은 다음을 포함하고 있습니다. • 순서대로 된 순서 값 (콜론과 공간이 뒤에 따라오는) • 컬럼 이름 • 순서 값 **(A)** 오름차순 **(D)** 내림차순	외부 순서 컬럼
OUTERJN	LEFT RIGHT	외부 조인 표시기
PARTCOLS	컬럼 이름	피연산자에 대한 파티션 컬럼
PREFETCH	LIST NONE SEQUENTIAL	적당한 사전 패치 유형.
RMTQTEXT	텍스트 조회	원격 조회 텍스트
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	행 잠금 인텐트.
ROWWIDTH	INTEGER	저장될 행의 넓이.
SCANDIR	FORWARD REVERSE	스캔 방향.
SCANGRAN	INTEGER	SCANUNITS에 표현된, 파티션 내 병렬 처리 또는 파티션 내 병행 스캔의 세분성.
SCANTYPE	LOCAL PARALLEL	파티션 내 병렬 처리, 색인 또는 테이블 스캔.

표 129. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
SCANUNIT	ROW PAGE	파티션 내 병렬 처리, 스캔 세분성 단위.
SERVER	원격 서버	원격 서버
SHARED	TRUE	파티션 내 병렬 처리, 공유 TEMP 표시기.
SLOWMAT	TRUE FALSE	느린 데이터화 플래그
SNGLPROD	TRUE FALSE	단일 에이전트에 의해 작성된 파티션 내 병렬 처리 정렬 또는 TEMP.
SORTKEY	이 유형의 각 행은 다음을 포함하고 있습니다. • 키 내의 순서 값 (콜론과 공간이 뒤에 따라오는) • 컬럼 이름 • 순서 값 **(A)** 오름차순 **(D)** 내림차순	정렬 키 컬럼.
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	파티션 내 병렬 처리, 정렬 유형.
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	테이블 잠금 인텐트.
TQDEGREE	INTEGER	파티션 내 병렬 처리, 테이블 대기행렬을 액세스하는 서브에이전트.
TQMERGE	TRUE FALSE	병합(저장) 테이블 대기행렬 표시기.
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	등록 정보를 읽는 테이블 대기행렬.
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	테이블 대기행렬 송신 등록 정보.

표 129. ARGUMENT_TYPE 및 ARGUMENT_VALUE 컬럼 값 (계속)

ARGUMENT_TYPE 값	가능한 ARGUMENT_VALUE 값	설명
TQTYPE	LOCAL	파티션 내 병렬 처리, 테이블 대기행렬.
TRUNCSRT	TRUE	절단 정렬(작성되는 행의 수를 제한).
UNIQUE	TRUE FALSE	고유 표시기.
UNIQKEY	이 유형의 각 행은 다음을 포함하고 있습니다. • 키 내의 순서 값(콜론과 공간이 뒤에 따라오는) • 컬럼 이름	고유 키 컬럼.
VOLATILE	TRUE	휘발성 테이블

EXPLAIN_INSTANCE 테이블

EXPLAIN_INSTANCE 테이블은 Explain 정보에 대한 주요한 제어 테이블입니다. Explain 테이블에서 데이터의 각 행은 이 테이블에 있는 한 개의 고유 행에 명시적으로 링크됩니다. EXPLAIN_INSTANCE 테이블은 Explain이 발생한 환경에 대한 정보뿐만 아니라, 설명된 SQL문의 자원에 대한 기본 정보도 제공합니다.

이 테이블의 정의에 대해서는 1479 페이지의 『EXPLAIN_INSTANCE 테이블 정의』에서 참조하십시오.

표 130. EXPLAIN_INSTANCE 테이블

컬럼 이름	데이터 유형	널(NULL) 키? 입력 가능?		설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	PK	이 Explain 요청의 개시자 권한 부여 ID.
EXPLAIN_TIME	TIMESTAMP	아니오	PK	Explain 요청에 대한 시작 시간.
SOURCE_NAME	VARCHAR(128)	아니오	PK	동적 명령문이 설명될 때 수행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름.
SOURCE_SCHEMA	VARCHAR(128)	아니오	PK	Explain 요청 소스의 스키마, 규정자.
EXPLAIN_OPTION	CHAR(1)	아니오	아니오	어떤 Explain 정보가 이 요청에 대해 필요한지를 나타냅니다. 가능한 값은 다음과 같습니다. **P** 계획 선택

표 130. EXPLAIN_INSTANCE 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
SNAPSHOT_TAKEN	CHAR(1)	아니오	아니오	Explain 스냅샷이 이 요청에 대해 응해야 할 지를 나타냅니다. 가능한 값은 다음과 같습니다. **Y** 예, Explain 스냅샷이 받아들여 EXPLAIN_STATEMENT 테이블에 저장되었습니다. 정규 Explain 정보가 또한 캡처됩니다. **N** Explain 스냅샷이 응하지 않습니다. 정규 Explain 정보가 캡처됩니다. **0** Explain 스냅샷만 응합니다. 정규 Explain 정보가 캡처되지 않습니다.
DB2_VERSION	CHAR(7)	아니오	아니오	설명 요청을 처리한 DB2 Universal Database에 대한 제품 릴리스. 양식은 vv.rr.m으로, 설명은 다음과 같습니다. **vv** 버전 번호 **rr** 릴리스 번호 **m** 유지보수 릴리스 번호
SQL_TYPE	CHAR(1)	아니오	아니오	Explain 인스턴스가 정적 또는 동적 SQL에 대한 것인지 나타냅니다. 가능한 값은 다음과 같습니다. **S** 정적 SQL **D** 동적 SQL
QUERYOPT	INTEGER	아니오	아니오	Explain 호출 시간에 SQL 컴파일러가 사용한 조회 최적화 클래스를 나타냅니다. 값은 SQL문을 설명하기 위해 SQL 컴파일러로 조회 최적화의 레벨이 수행한 것을 나타냅니다.
BLOCK	CHAR(1)	아니오	아니오	SQL문 컴파일 때 커서 블로킹의 유형이 사용한 것을 나타냅니다. SYSCAT.PACKAGES의 블록 컬럼에서 자세한 사항을 참조하십시오. 가능한 값은 다음과 같습니다. **N** 블로킹이 없음 **U** 명확한 커서 블록 **B** 모든 커서 블록

표 130. EXPLAIN_INSTANCE 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
ISOLATION	CHAR(2)	아니오	아니오	SQL문 컴파일 때 커서 분리의 유형이 사용한 것을 나타냅니다. SYSCAT.PACKAGES의 ISOLATION 컬럼에서 자세한 사항을 참조하십시오. 가능한 값은 다음과 같습니다. **RR** 반복 읽기(RR) **RS** 읽기 안정성(RS) **CS** 커서 안정성(CS) **UR** 미확약 읽기(UR)
BUFFPAGE	INTEGER	아니오	아니오	Explain 호출 시간에 BUFFPAGE 데이터베이스 구성의 설정 값을 나타냅니다.
AVG_APPLS	INTEGER	아니오	아니오	Explain 호출 시간에 AVG_APPLS 구성 매개변수의 값을 나타냅니다.
SORTHEAP	INTEGER	아니오	아니오	Explain 호출 시간에 SORTHEAP 데이터베이스 구성 설정 값을 나타냅니다.
LOCKLIST	INTEGER	아니오	아니오	Explain 호출 시간에 LOCKLIST 데이터베이스 구성 설정 값을 나타냅니다.
MAXLOCKS	SMALLINT	아니오	아니오	Explain 호출 시간에 MAXLOCKS 데이터베이스 구성 설정 값을 나타냅니다.
LOCKS_AVAIL	INTEGER	아니오	아니오	각 사용자를 위해 최적화 알고리즘에서 사용 가능하다고 가정되는 잠금의 번호를 포함합니다(LOCKLIST 및 MAXLOCKS에서 도출됨).
CPU_SPEED	DOUBLE	아니오	아니오	Explain 호출 시간에 CPUSPEED 데이터베이스 관리 프로그램 구성 설정 값을 나타냅니다.
REMARKS	VARCHAR(254)	예	아니오	사용자 제공 주석.
DBHEAP	INTEGER	아니오	아니오	Explain 호출 시간에 DBHEAP 데이터베이스 구성 설정 값을 나타냅니다.
COMM_SPEED	DOUBLE	아니오	아니오	Explain 호출 시간에 COMM_BANDWIDTH 데이터베이스 구성 설정 값을 나타냅니다.
PARALLELISM	CHAR(2)	아니오	아니오	가능한 값은 다음과 같습니다. • N = 병렬 처리 없음 • P = 파티션 내 병렬 처리 • IP = 파티션간 병렬 처리 • BP = 파티션 내 병렬 처리 및 파티션간 병렬 처리

표 130. EXPLAIN_INSTANCE 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
DATAJOINER	CHAR(1)	아니오	아니오	가능한 값은 다음과 같습니다. • N = 비 연합 시스템 플랜 • Y = 연합 시스템 플랜

EXPLAIN_OBJECT 테이블

EXPLAIN_OBJECT 테이블은 SQL문을 만족하기 위해 생성된 액세스 플랜이 요구한 데이터 오브젝트를 식별합니다.

이 테이블의 정의에 대해서는 1480 페이지의 『EXPLAIN_OBJECT 테이블 정의』에서 참조하십시오.

표 131. EXPLAIN_OBJECT 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	이 Explain 요청의 개시자 권한 부여 ID.
EXPLAIN_TIME	TIMESTAMP	아니오	FK	Explain 요청에 대한 시작 시간.
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적 명령문이 설명될 때 수행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름.
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청 소스의 스키마, 규정자.
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행과 관련한 Explain 정보의 레벨.
STMTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호.
OBJECT_SCHEMA	VARCHAR(128)	아니오	아니오	이 오브젝트가 속하는 스키마.
OBJECT_NAME	VARCHAR(128)	아니오	아니오	오브젝트 이름.
OBJECT_TYPE	CHAR(2)	아니오	아니오	오브젝트의 유형에 대한 설명적인 레이블.
CREATE_TIME	TIMESTAMP	예	아니오	오브젝트의 작성 시간; 테이블 함수인 경우 널(NULL).
STATISTICS_TIME	TIMESTAMP	예	아니오	오브젝트에 대한 통계 갱신 마지막 시간; 통계가 오브젝트에 대해 존재하지 않을 경우 널(NULL).
COLUMN_COUNT	SMALLINT	아니오	아니오	오브젝트의 컬럼 수.
ROW_COUNT	INTEGER	아니오	아니오	오브젝트의 평가 행의 수.
WIDTH	INTEGER	아니오	아니오	바이트로 오브젝트의 평균 길이. 색인에 대해 -1로 설정합니다.
PAGES	INTEGER	아니오	아니오	오브젝트가 버퍼 풀에서 차지한 평가 페이지 수. 테이블 함수에 대해 -1로 설정합니다.

표 131. EXPLAIN_OBJECT 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
DISTINCT	CHAR(1)	아니오	아니오	오브젝트의 행이 구별(예 중복 아님)인 경우를 나타냅니다. 가능한 값은 다음과 같습니다. **Y** 예 **N** 아니오
TABLESPACE_NAME	VARCHAR(128)	예	아니오	오브젝트가 저장된 테이블 공간 이름; 테이블 공간이 포함되어 있지 않은 경우 널(NULL)로 설정합니다.
OVERHEAD	DOUBLE	아니오	아니오	지정된 테이블 공간의 단일 무작위 입출력에 대해, 밀리세컨드로, 총 평가 오버헤드. 제어기 오버헤드, 디스크 탐색 그리고 대기 시간을 포함합니다. 테이블 공간이 포함되지 않을 경우 -1로 설정합니다.
TRANSFER_RATE	DOUBLE	아니오	아니오	지정된 테이블 공간에서, 밀리세컨드로, 데이터 페이지를 읽는 평가 시간. 테이블 공간이 포함되지 않을 경우 -1로 설정합니다.
PREFETCHSIZE	INTEGER	아니오	아니오	프리페치가 수행되기 전에 읽혀질 데이터 페이지의 수. 테이블 함수에 대해 -1로 설정합니다.
EXTENTSIZE	INTEGER	아니오	아니오	4K 데이터 페이지 내에, 확장 크기. 다음 컨테이너로 전환하기 전에 많은 페이지가 테이블 공간의 한 컨테이너에 기록됩니다. 테이블 함수에 대해 -1로 설정합니다.
CLUSTER	DOUBLE	아니오	아니오	색인으로 클러스터하는 데이터의 수준. >= 1, 이것은 CLUSTERRATIO입니다. >= 0 및 < 1, 이것은 CLUSTERFACTOR입니다. 통계가 사용 가능한 경우, 테이블, 테이블 함수에 대해 -1로 설정합니다.
NLEAF	INTEGER	아니오	아니오	색인 오브젝트의 값이 차지하는 잎 페이지 수. 통계가 사용 가능한 경우, 테이블, 테이블 함수에 대해 -1로 설정합니다.
NLEVELS	INTEGER	아니오	아니오	색인 오브젝트의 나무에서 색인 레벨의 번호. 통계가 사용 가능한 경우, 테이블, 테이블 함수에 대해 -1로 설정합니다.
FULLKEYCARD	BIGINT	아니오	아니오	색인 오브젝트에 포함된 구별 전체 키 값의 번호. 통계가 사용 가능한 경우, 테이블, 테이블 함수에 대해 -1로 설정합니다.
OVERFLOW	INTEGER	아니오	아니오	테이블에서 오버플로우 레코드의 총 수. 통계가 사용 가능하지 않은 경우, 테이블, 테이블 함수에 대해 -1로 설정합니다.
FIRSTKEYCARD	BIGINT	아니오	아니오	첫번째 구별 키 값의 수. 통계가 사용 가능한 경우, 테이블, 테이블 함수에 대해 -1로 설정합니다.

표 131. EXPLAIN_OBJECT 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
FIRST2KEYCARD	BIGINT	아니오	아니오	색인의 첫번째 {2,3,4} 컬럼을 사용한 구별 첫번째 키 값의 수. 통계가 사용 가능한 경우, 테이블 또는 테이블 함수에 대해 -1로 설정합니다.
FIRST3KEYCARD	BIGINT	아니오	아니오	
FIRST4KEYCARD	BIGINT	아니오	아니오	
SEQUENTIAL_PAGES	INTEGER	아니오	아니오	간격 사이에 큰 간격이 없거나 거의 없을 경우 색인 키 순서의 디스크에 지정된 리프 페이지의 수. 통계가 사용 가능한 경우, 테이블 또는 테이블 함수에 대해 -1로 설정합니다.
DENSITY	INTEGER	아니오	아니오	색인이 차지한 페이지 범위에서 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로서, 퍼센트로 표시됩니다 (0에서 100까지의 정수). 통계가 사용 가능한 경우, 테이블 또는 테이블 함수에 대해 -1로 설정합니다.

표 132. 가능한 OBJECT_TYPE 값

값	설명
IX	색인
TA	Table
TF	테이블 함수

EXPLAIN_OPERATOR 테이블

EXPLAIN_OPERATOR 테이블에는 SQL 컴파일러가 SQL문을 만족하는 데 필요한 모든 연산자가 나와 있습니다.

이 테이블의 정의에 대해서는 1481 페이지의 『EXPLAIN_OPERATOR 테이블 정의』에서 참조하십시오.

표 133. EXPLAIN_OPERATOR 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	이 Explain 요청의 개시자 권한 부여 ID.
EXPLAIN_TIME	TIMESTAMP	아니오	FK	Explain 요청에 대한 시작 시간.
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적 명령문이 설명될 때 수행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름.
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청 소스의 스키마, 규정자.
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행과 관련한 Explain 정보의 레벨.
STMTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호.

표 133. EXPLAIN_OPERATOR 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
SECTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호.
OPERATOR_ID	INTEGER	아니오	아니오	이 조회 내의 연산자에 대한 고유 ID.
OPERATOR_TYPE	CHAR(6)	아니오	아니오	연산자의 유형에 대한 설명적인 레이블.
TOTAL_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜 실행의 평가 누적 총 비용(timeron 단위).
IO_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜 실행의 평가 누적된 입출력 비용(데이터 페이지 입출력에서).
CPU_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜 실행의 평가 누적 CPU 비용(지시에서).
FIRST_ROW_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜에 대한 첫 행 페치의 평가 누적된 비용(timerons 단위). 이 값에는 요구한 처음의 오버헤드가 포함되어 있습니다.
RE_TOTAL_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜에 대한 마지막 행 페치의 평가 누적된 비용(timerons 단위).
RE_IO_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜에 대한 마지막 행 페치의 평가 누적된 입출력 비용(데이터 페이지 입출력에서).
RE_CPU_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜에 대한 마지막 행 페치의 평가 누적된 CPU 비용(timerons 단위).
COMM_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜 실행의 평가 누적된 통신 비용(TCP/IP 프레임에서).
FIRST_COMM_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜에 대한 첫 번째 행 페치의 평가 누적된 통신 비용(TCP/IP 프레임에서). 이 값에는 요구한 처음의 오버헤드가 포함되어 있습니다.
BUFFERS	DOUBLE	아니오	아니오	이 연산자 및 그 입력을 위한 예상 버퍼 요건.
REMOTE_TOTAL_COST	DOUBLE	아니오	아니오	원격 데이터베이스에서 조작 수행의 예상 누적 총 비용(timeron 단위).
REMOTE_COMM_COST	DOUBLE	아니오	아니오	이 연산자를 포함하기까지 선택된 액세스 플랜 실행의 예상 누적 통신 비용.

표 134. OPERATOR_TYPE 값

값	설명
DELETE	삭제
FETCH	페치
FILTER	필터 행
GENROW	생성 행
GRPBY	그룹 By

표 134. OPERATOR_TYPE 값 (계속)

값	설명
HSJOIN	해쉬 조인
INSERT	삽입
IXAND	동적 비트 맵핑 색인 ANDing
IXSCAN	색인 스캔
MSJOIN	병합 스캔 조인
NLJOIN	중첩 루프 조인
RETURN	결과
RIDSCN	행 식별자(RID) 스캔
RQUERY	원격 조회
SORT	정렬
TBSCAN	테이블 스캔
TEMP	임시 테이블 구조
TQ	테이블 대기행렬
UNION	단위
UNIQUE	중복 제거
UPDATE	갱신

EXPLAIN_PREDICATE 테이블

EXPLAIN_PREDICATE 테이블은 특정 연산자가 적용한 술어를 식별합니다.

이 테이블의 정의에 대해서는 1482 페이지의 『EXPLAIN_PREDICATE 테이블 정의』에서 참조하십시오.

표 135. EXPLAIN_PREDICATE 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	이 Explain 요청의 개시자 권한 부여 ID.
EXPLAIN_TIME	TIMESTAMP	아니오	FK	Explain 요청에 대한 시작 시간.
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적 명령문이 설명될 때 수행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름.
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청 소스의 스키마, 규정자.
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행과 관련한 Explain 정보의 레벨.
STMTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호.
OPERATOR_ID	INTEGER	아니오	아니오	이 조회 내의 연산자에 대한 고유 ID.

표 135. EXPLAIN_PREDICATE 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
PREDICATE_ID	INTEGER	아니오	아니오	지정된 연산자에 대한 술어의 고유 ID.
HOW_APPLIED	CHAR(5)	아니오	아니오	지정된 연산자가 술어를 사용하는 방법.
WHEN_EVALUATED	CHAR(3)	아니오	아니오	술어에 사용된 부속 조회가 평가되는 시간을 나타냅니다. 가능한 값은 다음과 같습니다. **공백** 이 술어에는 부속 조회가 포함되어 있지 않습니다. **EAA** 이 술어에 사용된 부속 조회는 응용프로그램(EAA)에서 평가됩니다. 이는 술어가 적용될 때, 특정 연산자가 처리한 모든 행에 대해 부속 조회가 재평가됨을 나타냅니다. **EAO** 이 술어에 사용된 부속 조회는 열기(EAO)에서 평가됩니다. 이는 부속 조회가 특정 연산자에 대해 한번만 재평가되며 결과가 각 행에 대한 술어의 응용프로그램에서 재사용됨을 나타냅니다. **MUL** 이 술어에 하나 이상의 부속 조회가 있습니다.
RELOP_TYPE	CHAR(2)	아니오	아니오	술어에 사용된 관계 연산자 유형.
SUBQUERY	CHAR(1)	아니오	아니오	부속 조회의 데이터 스트림이 술어를 필요로 하는지의 여부. 여러 부속 조회 스트림이 필요할 수 있습니다. 가능한 값은 다음과 같습니다. **N** 부속 조회 스트림이 필요하지 않습니다 **Y** 하나 이상의 부속 조회 스트림이 필요합니다.
FILTER_FACTOR	DOUBLE	아니오	아니오	술어가 규정할 행의 평가 소수.
PREDICATE_TEXT	CLOB(1M)	예	아니오	SQL문의 내부 표현에서 재작성된 술어의 텍스트. 사용 가능하지 않을 경우 널(NULL).

표 136. 가능한 HOW_APPLIED 값

값	설명
JOIN	조인 테이블에 사용됩니다
RESID	상주 술어로 평가됩니다
SARG	색인 또는 데이터 페이지에 대해 sargable 술어로 평가됩니다
START	시작 조건으로 사용됩니다

표 136. 가능한 HOW_APPLIED 값 (계속)

값	설명
STOP	종료 조건으로 사용됩니다

표 137. 가능한 RELOP_TYPE 값

값	설명
공백	적용 불가능
EQ	같음
GE	이상
GT	보다 큼
IN	목록에서
LE	이하
LK	같은
LT	보다 적음
NE	같지 않음
NL	널(NULL) 입니다
NN	널(NULL)이 아닙니다

EXPLAIN_STATEMENT 테이블

EXPLAIN_STATEMENT 테이블은 Explain 정보의 다른 레벨에 대해 존재할 때 SQL문의 텍스트를 포함합니다. 사용자가 입력한 대로 원래 SQL문은 SQL문을 만족시키기 위해 액세스 플랜을 선택하는 데 사용된 (최적화 알고리즘에 의해) 버전과 함께 테이블에 저장됩니다. 나중 버전은 재기록되고/거나 컴파일러가 결정한 대로 추가 술어로 향상되기 때문에 원래 것과 유사성이 거의 없습니다.

이 테이블의 정의에 대해서는, 1483 페이지의 『EXPLAIN_STATEMENT 테이블 정의』에서 참조하십시오.

표 138. EXPLAIN_STATEMENT 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	PK, FK	이 Explain 요청의 개시자 권한 부여 ID.
EXPLAIN_TIME	TIMESTAMP	아니오	PK, FK	Explain 요청에 대한 시작 시간.
SOURCE_NAME	VARCHAR(128)	아니오	PK, FK	동적 명령문이 설명될 때 수행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름.
SOURCE_SCHEMA	VARCHAR(128)	아니오	PK, FK	Explain 요청 소스의 스키마, 규정자.

표 138. EXPLAIN_STATEMENT 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
EXPLAIN_LEVEL	CHAR(1)	아니오	PK	이 행과 관련한 Explain 정보의 레벨. 가능한 값: **O** 원래 텍스트(사용자가 입력한 대로) **P** 계획 선택
STMTNO	INTEGER	아니오	PK	이 Explain 정보와 관련된 패키지 내의 명령문 번호. 동적 설명 SQL문에 대해 1로 설정하십시오. 정적 SQL문에 대해, 이 값은 SYSCAT.STATEMENTS 카탈로그 뷰에 대해 사용된 값과 동일합니다.
SECTNO	INTEGER	아니오	PK	SQL문을 포함하는 패키지 내의 섹션 번호. 동적 설명 SQL문에 대해, 이는 런타임 명령문에 대한 섹션을 보유하려고 사용된 섹션 번호입니다. 정적 SQL문에 대해, 이 값은 SYSCAT.STATEMENTS 카탈로그 뷰에 대해 사용된 값과 동일합니다.
QUERYNO	INTEGER	아니오	아니오	설명된 SQL문에 대한 숫자 식별자. CLP 또는 CLI를 통해서 실행된 동적 SQL문(EXPLAIN SQL문 제외)에 대해, 기본값은 순차적으로 증가된 값입니다. 그렇지 않으면, 기본값은 정적 SQL문에 대해 STMTNO의 값이고, 동적 SQL문에 대해 1입니다.
QUERYTAG	CHAR(20)	아니오	아니오	설명된 각 SQL문에 대한 식별자 태그. CLP를 통해 실행된 동적 SQL문에 대해(EXPLAIN SQL문 제외), 기본값은 'CLP'입니다. CLI를 통해 실행된 동적 SQL문에 대해(EXPLAIN SQL문 제외), 기본값은 'CLI'입니다. 그렇지 않으면, 사용된 기본값은 공백입니다.
STATEMENT_TYPE	CHAR(2)	아니오	아니오	설명되는 조회의 유형에 대한 설명적인 레이블. 가능한 값은 다음과 같습니다. **S** 선택 **D** 삭제 **DC** 현재 커서가 있는 곳을 삭제합니다 **I** 삽입 **U** 갱신 **UC** 현재 커서가 있는 곳을 갱신합니다

표 138. EXPLAIN_STATEMENT 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
UPDATABLE	CHAR(1)	아니오	아니오	명령문이 갱신 가능하다고 고려되는지 나타냅니다. 이는 잠재적으로 갱신 가능하다고 결정된 SELECT문에 특히 적합합니다. 가능한 값은 다음과 같습니다. ' ' 적용 불가능(공백) **N** 아니오 **Y** 예
DELETABLE	CHAR(1)	아니오	아니오	명령문이 삭제 가능하다고 고려되는지 나타냅니다. 이는 잠재적으로 삭제 가능하다고 결정된 SELECT문에 특히 적합합니다. 가능한 값은 다음과 같습니다. ' ' 적용 불가능(공백) **N** 아니오 **Y** 예
TOTAL_COST	DOUBLE	아니오	아니오	이 때에 액세스 플랜이 선택되지 않아서 EXPLAIN_LEVEL이 *O*이면 (원래 텍스트), 이 명령문에 대해 선택된 액세스 플랜 실행의 평가 총 비용.
STATEMENT_TEXT	CLOB(1M)	아니오	아니오	설명될 SQL문의 텍스트 부분 또는 텍스트. 설명의 플랜 선택 레벨에 대해 보여준 텍스트는 내부 표현으로 재구축되며 이후에 SQL 같이 됩니다 즉, 올바른 SQL 구문을 따르도록 재구축된 명령문을 보장할 수 없습니다.
SNAPSHOT	BLOB(10M)	예	아니오	Explain_Level이 보여준 SQL문에 대한 내부 표현의 스냅샷. 컬럼은 DB2 Visual Explain에 사용하기 위한 것입니다. 명령문의 특정 버전이 캡쳐될 때 선택된 액세스 플랜이 없으므로 EXPLAIN_LEVEL이 *0*이면(원래 명령문), 컬럼은 널(NULL)로 설정됩니다.
QUERY_DEGREE	INTEGER	아니오	아니오	Explain 호출시 파티션 내 병렬 처리의 수준을 나타냅니다. 원래 명령문에 대해, 이것은 파티션 내 병렬 처리의 올바른 수준을 포함합니다. PLAN SELECTION의 경우, 이것은 사용할 플랜에 대해 생성된 파티션 내 병렬 처리의 수준을 포함합니다.

EXPLAIN_STREAM 테이블

EXPLAIN_STREAM 테이블은 개의 연산자와 데이터 오브젝트간의 입력과 출력을 표시합니다. 데이터 오브젝트 그 자체가 EXPLAIN_OBJECT 테이블에 표시됩니다. 데이터 스트림과 관련된 연산자가 EXPLAIN_OPERATOR 테이블에 있습니다.

이 테이블의 정의에 대해서는 1484 페이지의 『EXPLAIN_STREAM 테이블 정의』에서 참조하십시오.

표 139. EXPLAIN_STREAM 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	FK	이 Explain 요청의 개시자 권한 부여 ID.
EXPLAIN_TIME	TIMESTAMP	아니오	FK	Explain 요청에 대한 시작 시간.
SOURCE_NAME	VARCHAR(128)	아니오	FK	동적 명령문이 설명될 때 수행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름.
SOURCE_SCHEMA	VARCHAR(128)	아니오	FK	Explain 요청 소스의 스키마, 규정자.
EXPLAIN_LEVEL	CHAR(1)	아니오	FK	이 행과 관련한 Explain 정보의 레벨.
STMTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	아니오	FK	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호.
STREAM_ID	INTEGER	아니오	아니오	지정된 연산자 내의 데이터 스트림에 대한 고유 ID.
SOURCE_TYPE	CHAR(1)	아니오	아니오	데이터 스트림의 소스를 나타냅니다: **O** 연산자 **D** 데이터 오브젝트
SOURCE_ID	SMALLINT	아니오	아니오	데이터 스트림의 소스인 이 조회에서 연산자에 대한 고유 ID. SOURCE_TYPE이 'D'인 경우, -1로 설정하십시오.
TARGET_TYPE	CHAR(1)	아니오	아니오	데이터 스트림의 목표를 나타냅니다: **O** 연산자 **D** 데이터 오브젝트
TARGET_ID	SMALLINT	아니오	아니오	데이터 스트림의 목표인 이 조회에서 연산자에 대한 고유 ID. TARGET_TYPE이 'D'인 경우, -1로 설정하십시오.
OBJECT_SCHEMA	VARCHAR(128)	예	아니오	영향을 준 데이터 오브젝트가 속해있는 스키마. SOURCE_TYPE 및 TARGET_TYPE이 'O'인 경우, 널(NULL)로 설정하십시오.

표 139. EXPLAIN_STREAM 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
OBJECT_NAME	VARCHAR(128)	예	아니오	데이터 스트림 주체의 오브젝트 이름. SOURCE_TYPE 및 TARGET_TYPE이 'O'인 경우, 널(NULL)로 설정하십시오.
STREAM_COUNT	DOUBLE	아니오	아니오	데이터 스트림의 평가 기본 행 수.
COLUMN_COUNT	SMALLINT	아니오	아니오	데이터 스트림 내의 컬럼 수.
PREDICATE_ID	INTEGER	아니오	아니오	이 스트림이 술어에 대한 부속 조회의 일부일 경우, 술어 ID를 여기에 반영합니다. 그렇지 않으면 컬럼은 -1로 설정됩니다.
COLUMN_NAMES	CLOB(1M)	예	아니오	이 컬럼에는 스트림과 관련된 컬럼의 오더 정보와 이름이 포함되어 있습니다. 이들 이름은 다음 형식으로 작성됩니다. `NAME1(A)+NAME2(D)+NAME3+NAME4` 여기서, *(A)*는 오름차순의 컬럼을 나타내고, *(D)*는 내림차순의 컬럼을 나타내며, 순서 정보가 없는 경우는 컬럼이 순서대로 되어 있지 않거나 순서에 상관없음을 나타냅니다.
PMID	SMALLINT	아니오	아니오	파티션 맵핑 ID.
SINGLE_NODE	CHAR(5)	예	아니오	이 데이터 스트림이 단일 파티션에 있는지 복수의 파티션에 있는지 나타냅니다. **MULT** 복수 파티션에서 **COOR** 조정자 노드에서 **HASH** 해쉬를 사용한 방향지정 **RID** 행 ID를 사용한 방향지정 **FUNC** 함수(PARTITION() 또는 NODENUMBER())를 사용한 방향지정 **CORR** 상관 값을 사용한 방향 지정 **숫자** 사전결정된 단일 노드로 방향지정됨
PARTITION_COLUMNS	CLOB(64K)	예	아니오	이 데이터 스트림이 파티션된 컬럼 목록.

ADVISE_INDEX 테이블

ADVISE_INDEX 테이블은 권장되는 색인을 나타냅니다.

이 테이블의 정의에 대해서는 1485 페이지의 『ADVISE_INDEX 테이블 정의』에서 참조하십시오.

표 140. ADVISE_INDEX 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
EXPLAIN_REQUESTER	VARCHAR(128)	아니오	아니오	이 Explain 요청의 개시자 권한 부여 ID.
EXPLAIN_TIME	TIMESTAMP	아니오	아니오	Explain 요청에 대한 시작 시간.
SOURCE_NAME	VARCHAR(128)	아니오	아니오	동적 명령문이 설명될 때 수행되는 패키지 이름 또는 정적 SQL이 설명될 때 소스 파일 이름.
SOURCE_SCHEMA	VARCHAR(128)	아니오	아니오	Explain 요청 소스의 스키마 또는 규정자.
EXPLAIN_LEVEL	CHAR(1)	아니오	아니오	이 행과 관련한 Explain 정보의 레벨.
STMTNO	INTEGER	아니오	아니오	이 Explain 정보와 관련된 패키지 내의 명령문 번호.
SECTNO	INTEGER	아니오	아니오	이 Explain 정보와 관련된 패키지 내의 명령문 섹션 번호.
QUERYNO	INTEGER	아니오	아니오	설명된 SQL문에 대한 숫자 식별자. CLP 또는 CLI를 통해서 실행된 동적 SQL문(EXPLAIN SQL문 제외)에 대해, 기본값은 순차적으로 증가된 값입니다. 그렇지 않으면, 기본값은 정적 SQL문에 대해 STMTNO의 값이고, 동적 SQL문에 대해 1입니다.
QUERYTAG	CHAR(20)	아니오	아니오	설명된 각 SQL문에 대한 식별자 태그. CLP를 통해 실행된 동적 SQL문에 대해(EXPLAIN SQL문 제외), 기본값은 'CLP'입니다. CLI를 통해 실행된 동적 SQL문에 대해(EXPLAIN SQL문 제외), 기본값은 'CLI'입니다. 그렇지 않으면, 사용된 기본값은 공백입니다.
NAME	VARCHAR(128)	아니오	아니오	색인 이름.
CREATOR	VARCHAR(128)	아니오	아니오	색인 이름의 규정자.
TBNAME	VARCHAR(128)	아니오	아니오	색인이 정의되는 테이블이나 별명의 이름.
TBCREATOR	VARCHAR(128)	아니오	아니오	테이블 이름의 규정자.
COLNAMES	CLOB(64K)	아니오	아니오	컬럼 이름 목록
UNIQUERULE	CHAR(1)	아니오	아니오	고유 규칙: D = 중복 허용 P = 1차 색인 U = 허용된 고유 항목만
COLCOUNT	SMALLINT	아니오	아니오	키 컬럼 수 + 포함 컬럼 수(있을 경우).
IID	SMALLINT	아니오	아니오	내부 색인 ID.
NLEAF	INTEGER	아니오	아니오	리프 페이지의 수. 통계를 내지 않았을 경우에는 -1
NLEVELS	SMALLINT	아니오	아니오	색인 레벨 수. 통계를 내지 않았을 경우에는 -1
FULLKEYCARD	BIGINT	아니오	아니오	구별되는 전체 키 값의 수. 통계를 내지 않았을 경우에는 -1.

표 140. ADVISE_INDEX 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
FIRSTKEYCARD	BIGINT	아니오	아니오	구별되는 첫번째 키 값의 수. 통계를 내지 않았을 경우에는 -1.
CLUSTERRATIO	SMALLINT	아니오	아니오	색인의 데이터 클러스터링 등급. 통계를 내지 않았거나 세부사항 색인 통계를 낸 경우에는 -1(이 경우에는 CLUSTERFACTOR가 대신 사용됩니다).
CLUSTERFACTOR	DOUBLE	아니오	아니오	클러스터링 수준의 더 정교한 측정 또는 상세 색인 통계가 수집되지 않았거나 색인이 별명에서 정의되는 경우에는 -1.
USERDEFINED	SMALLINT	아니오	아니오	사용자에 의해 정의.
SYSTEM_REQUIRED	SMALLINT	아니오	아니오	기본 키 또는 고유 키 제한조건에 이 색인이 필요하거나 이 색인이 입력된 테이블의 오브젝트 식별자(OID) 컬럼에 있는 색인인 경우에는 1입니다. 기본 키 또는 고유 키 제한조건에 이 색인이 필요하고 이 색인이 입력된 테이블의 오브젝트 식별자(OID) 컬럼에 있는 색인인 경우에는 2입니다. 그렇지 않으면 0.
CREATE_TIME	TIMESTAMP	아니오	아니오	색인이 작성된 시간.
STATS_TIME	TIMESTAMP	예	아니오	이 색인에 대한 기록된 통계를 마지막으로 변경할 때. 통계가 사용 가능하지 않으면 널(NULL).
PAGE_FETCH_PAIRS	VARCHAR(254)	아니오	아니오	문자 양식에서 표현된, 쌍으로 된 정수의 목록. 각 쌍은 가설 버퍼에 있는 페이지 수를 나타내며, 페이지 패치의 수는 가설 버퍼를 사용하여 이 색인으로 테이블을 스캔하는 데 필요합니다(사용 가능한 데이터가 없을 경우 문자열 길이 0).
REMARKS	VARCHAR(254)	예	아니오	사용자가 제공한 주석 또는 널(NULL).
DEFINER	VARCHAR(128)	아니오	아니오	색인을 작성한 사용자.
CONVERTED	CHAR(1)	아니오	아니오	이후 사용을 위해 예약됨.
SEQUENTIAL_PAGES	INTEGER	아니오	아니오	간격 사이에 큰 간격이 없거나 거의 없을 경우 색인 키 순서의 디스크에 지정된 리프 페이지의 수. (사용 가능한 통계가 없을 경우에는 -1).
DENSITY	INTEGER	아니오	아니오	색인이 차지한 페이지 범위에서 페이지 수에 대한 SEQUENTIAL_PAGES의 비율로서, 퍼센트로 표시됩니다(0에서 100까지의 정수로 표시되고, 사용 가능한 통계가 없을 경우에는 -1).
FIRST2KEYCARD	BIGINT	아니오	아니오	색인의 처음 두 컬럼을 사용하는 구별 키의 수(통계를 내지 않았거나 적용 불가능한 경우에는 -1)
FIRST3KEYCARD	BIGINT	아니오	아니오	색인의 처음 세 컬럼을 사용하는 구별 키의 수(통계를 내지 않았거나 적용 불가능한 경우에는 -1)

표 140. ADVISE_INDEX 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
FIRST4KEYCARD	BIGINT	아니오	아니오	색인의 처음 네 컬럼을 사용하는 구별 키의 수(통계를 내지 않았거나 적용 불가능한 경우에는 -1)
PCTFREE	SMALLINT	아니오	아니오	처음 색인을 구축하는 동안 예약될 각 색인 리프 페이지의 비율. 이 공간은 색인이 만들어진 후에 삽입될 경우를 대비하여 제공됩니다.
UNIQUE_COLCOUNT	SMALLINT	아니오	아니오	고유 키에 필요한 컬럼 수. 항상 <=COLCOUNT. include 컬럼이 있을 경우에만 < COLCOUNT입니다. 색인에 고유 키(중복 허용)가 없을 경우에는 -1입니다.
MINPCTUSED	SMALLINT	아니오	아니오	0이 아니라면, 온라인 색인 재구성이 작동 가능화되며 값은 페이지 병합 전의 최소 사용 공간의 임계값입니다.
REVERSE_SCANS	CHAR(1)	아니오	아니오	Y = 색인이 역 스캔을 지원함 N = 색인이 역 스캔을 지원하지 않음
USE_INDEX	CHAR(1)	예	아니오	Y = 색인이 권장되거나 평가됨 N = 색인이 권장되지 않음
CREATION_TEXT	CLOB(1M)	아니오	아니오	SQL문이 색인을 작성한 적이 있습니다.
PACKED_DESC	BLOB(20M)	예	아니오	테이블의 내부 설명.

ADVISE_WORKLOAD 테이블

ADVISE_WORKLOAD 테이블은 워크로드를 구성하는 명령문을 나타냅니다. 워크로드에 대한 *관리 안내서: 성능*에서 자세한 내용을 참조하십시오.

이 테이블의 정의에 대해서는 1487 페이지의 『ADVISE_WORKLOAD 테이블 정의』에서 참조하십시오.

표 141. ADVISE_WORKLOAD 테이블

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
WORKLOAD_NAME	CHAR(128)	아니오	아니오	이 명령문이 속하는 SQL문 콜렉션(워크로드)의 이름.
STATEMENT_NO	INTEGER	아니오	아니오	이 설명 정보가 관련되는 워크로드 내의 명령문 번호.
STATEMENT_TEXT	CLOB(1M)	아니오	아니오	SQL문의 내용.
STATEMENT_TAG	VARCHAR(256)	아니오	아니오	설명된 각 SQL문에 대한 식별자 태그.
FREQUENCY	INTEGER	아니오	아니오	이 명령문이 워크로드 내에 나타나는 횟수.
IMPORTANCE	DOUBLE	아니오	아니오	명령문의 중요도.

표 141. ADVISE_WORKLOAD 테이블 (계속)

컬럼 이름	데이터 유형	널(NULL) 입력 가능?	키?	설명
COST_BEFORE	DOUBLE	예	아니오	권장되는 색인이 작성되지 않는 경우 조회의 비용(timerons 단위).
COST_AFTER	DOUBLE	예	아니오	권장되는 색인이 작성되는 경우 조회의 비용(timerons 단위).

Explain 테이블에 대한 테이블 정의

Explain을 호출하려면 Explain 테이블을 먼저 작성해야 합니다. 다음 정의는 필요한 Explain 테이블을 작성하는 방법을 지정합니다.

- 1478 페이지의 『EXPLAIN_ARGUMENT 테이블 정의』
- 1479 페이지의 『EXPLAIN_INSTANCE 테이블 정의』
- 1480 페이지의 『EXPLAIN_OBJECT 테이블 정의』
- 1481 페이지의 『EXPLAIN_OPERATOR 테이블 정의』
- 1482 페이지의 『EXPLAIN_PREDICATE 테이블 정의』
- 1483 페이지의 『EXPLAIN_STATEMENT 테이블 정의』
- 1484 페이지의 『EXPLAIN_STREAM 테이블 정의』
- 1485 페이지의 『ADVISE_INDEX 테이블 정의』
- 1487 페이지의 『ADVISE_WORKLOAD 테이블 정의』

또는 'sqllib' 디렉토리의 'misc' 서브디렉토리에 있는 EXPLAIN.DDL 파일에 제공된 샘플 명령행 프로세서 입력 스크립트를 사용하여 작성하십시오. Explain 테이블이 필요한 데이터베이스와 연결하십시오. 그런 다음, `db2 -tf EXPLAIN.DDL` 명령 및 작성될 테이블을 발행하십시오.

EXPLAIN_ARGUMENT 테이블 정의

```
CREATE TABLE EXPLAIN_ARGUMENT ( EXPLAIN_REQUESTER   VARCHAR(128)  NOT NULL,
                                EXPLAIN_TIME        TIMESTAMP     NOT NULL,
                                SOURCE_NAME         VARCHAR(128)  NOT NULL,
                                SOURCE_SCHEMA       VARCHAR(128)  NOT NULL,
                                EXPLAIN_LEVEL       CHAR(1)       NOT NULL,
                                STMTNO              INTEGER       NOT NULL,
                                SECTNO              INTEGER       NOT NULL,
                                OPERATOR_ID         INTEGER       NOT NULL,
                                ARGUMENT_TYPE       CHAR(8)       NOT NULL,
                                ARGUMENT_VALUE      VARCHAR(1024) NOT NULL,
                                LONG_ARGUMENT_VALUE CLOB(1M)      NOT LOGGED,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

EXPLAIN_INSTANCE 테이블 정의

```
CREATE TABLE EXPLAIN_INSTANCE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_OPTION    CHAR(1)      NOT NULL,
                                SNAPSHOT_TAKEN    CHAR(1)      NOT NULL,
                                DB2_VERSION       CHAR(7)      NOT NULL,
                                SQL_TYPE          CHAR(1)      NOT NULL,
                                QUERYOPT          INTEGER      NOT NULL,
                                BLOCK             CHAR(1)      NOT NULL,
                                ISOLATION         CHAR(2)      NOT NULL,
                                BUFFPAGE          INTEGER      NOT NULL,
                                AVG_APPLS         INTEGER      NOT NULL,
                                SORTHEAP          INTEGER      NOT NULL,
                                LOCKLIST          INTEGER      NOT NULL,
                                MAXLOCKS          SMALLINT     NOT NULL,
                                LOCKS_AVAIL       INTEGER      NOT NULL,
                                CPU_SPEED         DOUBLE       NOT NULL,
                                REMARKS           VARCHAR(254),
                                DBHEAP            INTEGER      NOT NULL,
                                COMM_SPEED        DOUBLE       NOT NULL,
                                PARALLELISM       CHAR(2)      NOT NULL,
                                DATAJOINER        CHAR(1)      NOT NULL,
                                     PRIMARY KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                    SOURCE_SCHEMA))
```

EXPLAIN_OBJECT 테이블 정의

```
CREATE TABLE EXPLAIN_OBJECT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                              EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                              STMTNO            INTEGER      NOT NULL,
                              SECTNO            INTEGER      NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128) NOT NULL,
                              OBJECT_NAME       VARCHAR(128) NOT NULL,
                              OBJECT_TYPE       CHAR(2)      NOT NULL,
                              CREATE_TIME       TIMESTAMP,
                              STATISTICS_TIME   TIMESTAMP,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              ROW_COUNT         INTEGER      NOT NULL,
                              WIDTH             INTEGER      NOT NULL,
                              PAGES             INTEGER      NOT NULL,
                              DISTINCT          CHAR(1)      NOT NULL,
                              TABLESPACE_NAME   VARCHAR(128),
                              OVERHEAD          DOUBLE       NOT NULL,
                              TRANSFER_RATE     DOUBLE       NOT NULL,
                              PREFETCHSIZE      INTEGER      NOT NULL,
                              EXTENTSIZE        INTEGER      NOT NULL,
                              CLUSTER           DOUBLE       NOT NULL,
                              NLEAF             INTEGER      NOT NULL,
                              NLEVELS           INTEGER      NOT NULL,
                              FULLKEYCARD       BIGINT       NOT NULL,
                              OVERFLOW          INTEGER      NOT NULL,
                              FIRSTKEYCARD      BIGINT       NOT NULL,
                              FIRST2KEYCARD     BIGINT       NOT NULL,
                              FIRST3KEYCARD     BIGINT       NOT NULL,
                              FIRST4KEYCARD     BIGINT       NOT NULL,
                              SEQUENTIAL_PAGES  INTEGER      NOT NULL,
                              DENSITY           INTEGER      NOT NULL,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

EXPLAIN_OPERATOR 테이블 정의

```
CREATE TABLE EXPLAIN_OPERATOR ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                STMTNO            INTEGER      NOT NULL,
                                SECTNO            INTEGER      NOT NULL,
                                OPERATOR_ID       INTEGER      NOT NULL,
                                OPERATOR_TYPE     CHAR(6)      NOT NULL,
                                TOTAL_COST        DOUBLE       NOT NULL,
                                IO_COST           DOUBLE       NOT NULL,
                                CPU_COST          DOUBLE       NOT NULL,
                                FIRST_ROW_COST    DOUBLE       NOT NULL,
                                RE_TOTAL_COST     DOUBLE       NOT NULL,
                                RE_IO_COST        DOUBLE       NOT NULL,
                                RE_CPU_COST       DOUBLE       NOT NULL,
                                COMM_COST         DOUBLE       NOT NULL,
                                FIRST_COMM_COST   DOUBLE       NOT NULL,
                                REMOTE_TOTAL_COST DOUBLE       NOT NULL,
                                REMOTE_COMM_COST  DOUBLE       NOT NULL,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

EXPLAIN_PREDICATE 테이블 정의

```
CREATE TABLE EXPLAIN_PREDICATE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                 EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                 STMTNO            INTEGER      NOT NULL,
                                 SECTNO            INTEGER      NOT NULL,
                                 OPERATOR_ID       INTEGER      NOT NULL,
                                 PREDICATE_ID      INTEGER      NOT NULL,
                                 HOW_APPLIED       CHAR(5)      NOT NULL,
                                 WHEN_EVALUATED    CHAR(3)      NOT NULL,
                                 RELOP_TYPE        CHAR(2)      NOT NULL,
                                 SUBQUERY          CHAR(1)      NOT NULL,
                                 FILTER_FACTOR     DOUBLE       NOT NULL,
                                 PREDICATE_TEXT    CLOB(1M)     NOT LOGGED,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

EXPLAIN_STATEMENT 테이블 정의

```
CREATE TABLE EXPLAIN_STATEMENT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                 EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                 STMTNO            INTEGER      NOT NULL,
                                 SECTNO            INTEGER      NOT NULL,
                                 QUERYNO           INTEGER      NOT NULL,
                                 QUERYTAG          CHAR(20)     NOT NULL,
                                 STATEMENT_TYPE    CHAR(2)      NOT NULL,
                                 UPDATABLE         CHAR(1)      NOT NULL,
                                 DELETABLE         CHAR(1)      NOT NULL
                                 TOTAL_COST        DOUBLE       NOT NULL,
                                 STATEMENT_TEXT    CLOB(1M)     NOT NULL
                                                                NOT LOGGED,
                                 SNAPSHOT          BLOB(10M)    NOT LOGGED,
                                 QUERY_DEGREE      INTEGER      NOT NULL,
                                     PRIMARY KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA,
                                                  EXPLAIN_LEVEL,
                                                  STMTNO,
                                                     SECTNO),
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                     SOURCE_SCHEMA)
                                     REFERENCES EXPLAIN_INSTANCE
                                  ON DELETE CASCADE )
```

EXPLAIN_STREAM 테이블 정의

```
CREATE TABLE EXPLAIN_STREAM ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                              EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                              STMTNO            INTEGER      NOT NULL,
                              SECTNO            INTEGER      NOT NULL,
                              STREAM_ID         INTEGER      NOT NULL,
                              SOURCE_TYPE       CHAR(1)      NOT NULL,
                              SOURCE_ID         SMALLINT     NOT NULL,
                              TARGET_TYPE       CHAR(1)      NOT NULL,
                              TARGET_ID         SMALLINT     NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128),
                              OBJECT_NAME       VARCHAR(128),
                              STREAM_COUNT      DOUBLE       NOT NULL,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              PREDICATE_ID      INTEGER      NOT NULL,
                              COLUMN_NAMES      CLOB(1M)     NOT LOGGED,
                              PMID              SMALLINT     NOT NULL,
                              SINGLE_NODE       CHAR(5),
                              PARTITION_COLUMNS CLOB(64K)    NOT LOGGED,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

ADVISE_INDEX 테이블 정의

```
CREATE TABLE ADVISE_INDEX (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           EXPLAIN_TIME      TIMESTAMP    NOT NULL
                             WITH DEFAULT CURRENT TIMESTAMP,
                           SOURCE_NAME       VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           SOURCE_SCHEMA     VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           EXPLAIN_LEVEL     CHAR(1)      NOT NULL
                             WITH DEFAULT '',
                           STMTNO            INTEGER      NOT NULL
                             WITH DEFAULT 0,
                           SECTNO            INTEGER      NOT NULL
                             WITH DEFAULT 0,
                           QUERYNO           INTEGER      NOT NULL
                             WITH DEFAULT 0,
                           QUERYTAG          CHAR(20)     NOT NULL
                             WITH DEFAULT '',
                           NAME              VARCHAR(128) NOT NULL,
                           CREATOR           VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           TBNAME            VARCHAR(128) NOT NULL,
                           TBCREATOR         VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           COLNAMES          CLOB(64K)    NOT NULL,
                           UNIQUERULE        CHAR(1)      NOT NULL
                             WITH DEFAULT '',
                           COLCOUNT          SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           IID               SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           NLEAF             INTEGER      NOT NULL
                             WITH DEFAULT 0,
                           NLEVELS           SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           FIRSTKEYCARD      BIGINT       NOT NULL
                             WITH DEFAULT 0,
                           FULLKEYCARD       BIGINT       NOT NULL
                             WITH DEFAULT 0,
                           CLUSTERRATIO      SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           CLUSTERFACTOR     DOUBLE       NOT NULL
                             WITH DEFAULT 0,
                           USERDEFINED       SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           SYSTEM_REQUIRED   SMALLINT     NOT NULL
                             WITH DEFAULT 0,
                           CREATE_TIME       TIMESTAMP    NOT NULL
                             WITH DEFAULT CURRENT TIMESTAMP,
                           STATS_TIME        TIMESTAMP
                             WITH DEFAULT CURRENT TIMESTAMP,
                           PAGE_FETCH_PAIRS  VARCHAR(254) NOT NULL
                             WITH DEFAULT '',
                           REMARKS           VARCHAR(254)
                             WITH DEFAULT '',
                           DEFINER           VARCHAR(128) NOT NULL
                             WITH DEFAULT '',
                           CONVERTED          CHAR(1) NOT NULL
                             WITH DEFAULT '',
```

```
SEQUENTIAL_PAGES  INTEGER      NOT NULL
                     WITH DEFAULT 0,
DENSITY           INTEGER      NOT NULL
                     WITH DEFAULT 0,
FIRST2KEYCARD     BIGINT       NOT NULL
                     WITH DEFAULT 0,
FIRST3KEYCARD     BIGINT       NOT NULL
                     WITH DEFAULT 0,
FIRST4KEYCARD     BIGINT       NOT NULL
                     WITH DEFAULT 0,
PCTFREE           SMALLINT     NOT NULL
                     WITH DEFAULT -1,
UNIQUE_COLCOUNT   SMALLINT     NOT NULL
                     WITH DEFAULT -1,
MINPCTUSED        SMALLINT     NOT NULL
                     WITH DEFAULT 0,
REVERSE_SCANS     CHAR(1)      NOT NULL
                     WITH DEFAULT 'N',
USE_INDEX         CHAR(1),
CREATION_TEXT     CLOB(1M)     NOT NULL
                     NOT LOGGED WITH DEFAULT '',
PACKED_DESC       BLOB(1M)     NOT LOGGED)
```

ADVISE_WORKLOAD 테이블 정의

```
CREATE TABLE ADVISE_WORKLOAD (WORKLOAD_NAME  CHAR(128)    NOT NULL
                                WITH DEFAULT 'WK0',
                              STATEMENT_NO   INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              STATEMENT_TEXT CLOB(1M)     NOT NULL NOT LOGGED,
                              STATEMENT_TAG  VARCHAR(256) NOT NULL
                                WITH DEFAULT '',
                              FREQUENCY      INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              IMPORTANCE     DOUBLE       NOT NULL
                                WITH DEFAULT 1,
                              COST_BEFORE    DOUBLE,
                              COST_AFTER     DOUBLE)
```

부록L. Explain 레지스터 값

이 부록에서는 CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터 값 서로간의 상호작용과 PREP 및 BIND 명령과의 상호작용에 대해 설명합니다.

CURRENT EXPLAIN MODE 및 CURRENT EXPLAIN SNAPSHOT 특수 레지스터 값은 동적 SQL에 대해 다음과 같이 상호작용합니다.

표 142. 동적 SQL에 대한 Explain 특수 레지스터 값의 상호작용

EXPLAIN SNAPSHOT 값	EXPLAIN MODE 값				
	NO	**YES**	**EXPLAIN**	**RECOMMEND INDEXES**	**EVALUATE INDEXES**
NO	• 리턴되는 조회의 결과.	• 데이터가 들어 있는 Explain 테이블. • 리턴되는 조회의 결과.	• 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).	• 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인이 권장됨.	• 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인이 평가됨.
YES	• Explain 스냅샷. • 리턴되는 조회의 결과.	• 데이터가 들어 있는 Explain 테이블. • Explain 스냅샷. • 리턴되는 조회의 결과.	• 데이터가 들어 있는 Explain 테이블. • Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).	• 데이터가 들어 있는 Explain 테이블. • Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인이 권장됨.	• 데이터가 들어 있는 Explain 테이블. • Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인이 평가됨.

표 142. 동적 SQL에 대한 Explain 특수 레지스터 값의 상호작용 (계속)

EXPLAIN SNAPSHOT 값	EXPLAIN MODE 값				
	NO	YES	EXPLAIN	RECOMMEND INDEXES	EVALUATE INDEXES
EXPLAIN	• Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).	• 데이터가 들어 있는 Explain 테이블. • Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).	• 데이터가 들어 있는 Explain 테이블. • Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).	• 데이터가 들어 있는 Explain 테이블. • Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인이 권장됨.	• 데이터가 들어 있는 Explain 테이블. • Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인이 평가됨.

CURRENT EXPLAIN MODE 특수 레지스터는 동적 SQL에 대해 다음과 같이 EXPLAIN 바인드 옵션과 상호작용합니다.

표 143. EXPLAIN 바인드 옵션 및 CURRENT EXPLAIN MODE의 상호작용

EXPLAIN MODE 값	EXPLAIN 바인드 옵션 값		
	NO	YES	ALL
NO	• 리턴되는 조회의 결과.	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되는 조회의 결과.	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되는 조회의 결과.
YES	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되는 조회의 결과.	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되는 조회의 결과.	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되는 조회의 결과.
EXPLAIN	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).

표 143. EXPLAIN 바인드 옵션 및 CURRENT EXPLAIN MODE의 상호작용 (계속)

EXPLAIN MODE 값	EXPLAIN 바인드 옵션 값		
	NO	YES	ALL
RECOMMEND INDEXES	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인 권장	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인 권장	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인 권장
EVALUATE INDEXES	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인 평가	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인 평가	• 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 동적 SQL에 대한 데이터가 들어 있는 Explain 테이블. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문). • 색인 평가

CURRENT EXPLAIN SNAPSHOT 특수 레지스터는 동적 SQL에 대해 다음과 같이 EXPLSNAP 바인드 옵션과 상호작용합니다.

표 144. EXPLSNAP 바인드 옵션 및 CURRENT EXPLAIN SNAPSHOT의 상호작용

EXPLAIN SNAPSHOT 값	EXPLSNAP 바인드 옵션 값		
	NO	YES	ALL
NO	• 리턴되는 조회의 결과.	• 정적 SQL에 대한 Explain 스냅샷. • 리턴되는 조회의 결과.	• 정적 SQL에 대한 Explain 스냅샷. • 동적 SQL에 대한 Explain 스냅샷. • 리턴되는 조회의 결과.
YES	• 동적 SQL에 대한 Explain 스냅샷. • 리턴되는 조회의 결과.	• 정적 SQL에 대한 Explain 스냅샷. • 동적 SQL에 대한 Explain 스냅샷. • 리턴되는 조회의 결과.	• 정적 SQL에 대한 Explain 스냅샷. • 동적 SQL에 대한 Explain 스냅샷. • 리턴되는 조회의 결과.

표 144. EXPLSNAP 바인드 옵션 및 CURRENT EXPLAIN SNAPSHOT의 상호작용 (계속)

EXPLAIN SNAPSHOT 값	EXPLSNAP 바인드 옵션 값		
	NO	**YES**	**ALL**
EXPLAIN	• 동적 SQL에 대한 Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).	• 정적 SQL에 대한 Explain 스냅샷. • 동적 SQL에 대한 Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).	• 정적 SQL에 대한 Explain 스냅샷. • 동적 SQL에 대한 Explain 스냅샷. • 리턴되지 않는 조회의 결과(실행되지 않은 동적 명령문).

부록M. 순환 예: 부품표(BOM: Bill of Materials)

BOM 응용프로그램은 많은 비지니스 환경에서의 공통 요구사항입니다. BOM 응용프로그램에 대한 순환 공통 테이블 표현식의 성능을 표시하려면, 부품에 필요한 부속 제품 수량과 관련 부속 부분을 가진 부품표를 참고하십시오. 이 예의 경우, 다음과 같이 테이블을 작성합니다.

```
CREATE TABLE PARTLIST
              (PART VARCHAR(8),
               SUBPART VARCHAR(8),
               QUANTITY INTEGER);
```

이 예에 대한 조회 결과를 제공하기 위해, PARTLIST 테이블에 다음 값들이 들어 있다고(populate) 가정합니다.

부품	부속 부품	수량
00	01	5
00	05	3
01	02	2
01	03	3
01	04	4
01	06	3
02	05	7
02	06	6
03	07	6
04	08	10
04	09	11
05	10	10
05	11	10
06	12	10
06	13	10
07	14	8
07	12	8

예 1: 단일 레벨 전개

첫번째 예는 단일 레벨 전개라고 합니다. 『'01'로 식별되는 부품 구축에 필요한 부품은 무엇인가?』란 질문에 대답을 합니다. 이 목록에는 직접적인 부속 부품, 부속 부품의 부속 부품 등이 포함됩니다. 그러나, 한 부품이 여러 번 사용되면 그 부속 부품은 한 번만 나열됩니다.

```
WITH RPL (PART, SUBPART, QUANTITY) AS
     (  SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
          FROM PARTLIST ROOT
          WHERE ROOT.PART = '01'
       UNION ALL
        SELECT CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
          FROM RPL PARENT, PARTLIST CHILD
          WHERE PARENT.SUBPART = CHILD.PART
     )
SELECT DISTINCT PART, SUBPART, QUANTITY
 FROM RPL
  ORDER BY PART, SUBPART, QUANTITY;
```

상기 조회에는 공통 테이블 표현식이 포함되는데, 이 표현식은 이 조회의 순환적 부분을 표시하는 *RPL*이란 이름으로 식별됩니다. 이는 순환 공통 테이블 표현식의 기본 요소를 예합니다.

*초기설정 fullselect*로 명명되는 UNION의 첫번째 fullselect는 부품 '01'에 대한 직접 자를 구합니다. 이 fullselect의 FROM절은 원시 테이블을 참조하지, 그 자체(이 경우 *RPL*)를 참조하지 않습니다. 이 첫번째 fullselect의 결과는 공통 테이블 표현식 *RPL*(순환 PARTLIST)로 갑니다. 이 예에서 처럼, UNION은 항상 UNION ALL입니다.

UNION의 두 번째 fullselect는 *RPL*을 사용하여 부속 부품의 부속 부품을 계산합니다. 이 때 FROM절은 공통 테이블 표현식 *RPL*을 참조하고, 원시 테이블의 부품을 현재 결과의 부속 부품에 연결시키는 원시 테이블은 *RPL*에 들어 있습니다. 그 결과는 다시 *RPL*로 갑니다. 그런 후 UNION의 두 번째 피연산자가 하위 부품이 더 이상 없을 때까지 반복적으로 사용됩니다.

이 조회의 주요 fullselect에 있는 SELECT DISTINCT는 같은 부품/부속 부품이 한 번 이상 나열되지 않도록 합니다.

조회의 결과는 다음과 같습니다.

```
부품     부속 부품     수량
-------- -------- -----------
01       02                 2
01       03                 3
01       04                 4
01       06                 3
02       05                 7
02       06                 6
03       07                 6
04       08                10
04       09                11
05       10                10
05       11                10
06       12                10
06       13                10
07       12                 8
07       14                 8
```

부품 '01'로부터 '02'로 가면 다시 '06'으로 가는 식으로 계속되는 결과를 관찰하십시오. 또한, 부품 '01'을 통해 직접적으로 한 번, 다음 번엔 '02'를 통해서 부품 '06'이 두 번 도달함에 유의하십시오. 그러나, 출력시에 그 부속 구성요소들은 한 번만 나열됩니다.(이는 SELECT DISTINCT를 사용한 결과입니다.)

순환 공통 테이블 표현식을 가지고 무한 루프를 도입하는 것이 가능하다는 사실을 기억하는 것이 중요합니다. 이 예에서, 무한 루프는 상위 테이블과 하위 테이블을 조인시키는 두 번째 피연산자의 검색 조건이 다음과 같이 코드화된 경우, 무한 루프가 생길 수 있습니다.

```
PARENT.SUBPART = CHILD.SUBPART
```

무한 루프를 만드는 이 예는 의도한대로 작성하지 못한 경우임이 분명합니다. 그러나, 순환 주기가 명백히 종료되도록 하려면 작성될 것이 무엇인지를 결정하는 데 있어서도 주의를 기울여야 합니다.

이 예에서 산출되는 결과는 순환 공통 테이블 표현식을 사용하지 않고 응용프로그램에서 산출될 수 있습니다. 그러나, 이러한 접근방법은 모든 레벨의 순환에 대해 새로 조회를 시작해야 합니다. 또한, 응용프로그램은 결과의 순서를 정하기 위해 데이터베이스에 모든 결과들을 다시 두어야 합니다. 이러한 접근방식은 응용프로그

램 논리를 복잡하게 하고 제대로 수행되지 않습니다. 응용프로그램 논리는 요약 및 들여쓰기된 전개 조회와 같은 다른 BOM 조회의 경우 훨씬 어렵고 비효율적입니다.

예 2: 요약 전개(explosion)

두 번째 예는 요약 전개입니다. 여기에서 제기되는 문제는 부품 '01'을 만드는 데 필요한 각 부품의 전체 수량은 얼마인가 하는 것입니다. 단일 레벨 전개와의 가장 큰 차이점은 수량을 총계할 필요가 있는 것이라고 할 수 있습니다. 첫번째 예는 필요할 때마다 부품에 필요한 부속 부품의 수량을 표시합니다. 여기에는 '01' 부품을 만드는 데 얼마나 많은 부속 부품이 필요한지는 보여주지 않습니다.

```
WITH RPL (PART, SUBPART, QUANTITY) AS
   (
      SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
          FROM PARTLIST ROOT
          WHERE ROOT.PART = '01'
       UNION ALL
      SELECT PARENT.PART, CHILD.SUBPART, PARENT.QUANTITY*CHILD.QUANTITY
          FROM RPL PARENT, PARTLIST CHILD
          WHERE PARENT.SUBPART = CHILD.PART
   )
SELECT PART, SUBPART, SUM(QUANTITY) AS "Total QTY Used"
 FROM RPL
  GROUP BY PART, SUBPART
  ORDER BY PART, SUBPART;
```

위에 나온 조회에서, *RPL*로 식별되는 순환 공통 테이블 표현식에서 UNION의 두 번째 피연산자 선택 목록은 수량 총계를 보여줍니다. 얼마나 많은 부속 부품이 사용되는지를 알아 보려면, 상위 부품 수량을 하위 부품의 상위 부품당 수량으로 곱합니다. 다른 곳에서 한 부품이 여러 번 사용된 경우, 또 다른 최종 총계가 필요합니다. 이는 공통 테이블 표현식 *RPL*을 그룹화하고 주요 fullselect의 선택 목록에 있는 SUM 컬럼 함수를 사용함으로써 이루어집니다.

조회의 결과는 다음과 같습니다.

```
부품     부속 부품     총 사용량
-------- -------- --------------
01       02                    2
01       03                    3
01       04                    4
01       05                   14
01       06                   15
```

```
01       07                 18
01       08                 40
01       09                 44
01       10                140
01       11                140
01       12                294
01       13                150
01       14                144
```

출력을 보면서 부속 부품 '06' 행을 살펴보십시오. 15의 총 사용량 값은 부품 '01'의 수량 3과 부품 '01'에 두 번 필요한 부품 '02'의 수량에서 도출된 것입니다.

예 3: 깊이 제어

조회에 대해 관심 있는 테이블 내에 더 많은 부품 레벨이 있을 때 어떤 일이 발생하는 지에 관한 질문을 생각해 봅시다. 다시 말해, 『'01'로 식별되는 부품을 만드는 데 필요한 첫번째 두 레벨의 부품은 무엇인가』라는 질문에 대답하기 위한 조회는 어떻게 작성됩니까? 예의 명료성을 위해 레벨이 결과에 포함됩니다.

```
WITH RPL (LEVEL, PART, SUBPART, QUANTITY) AS
      (
         SELECT 1,               ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
          FROM PARTLIST ROOT
          WHERE ROOT.PART = '01'
       UNION ALL
         SELECT PARENT.LEVEL+1, CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
          FROM RPL PARENT, PARTLIST CHILD
          WHERE PARENT.SUBPART = CHILD.PART
            AND PARENT.LEVEL < 2
      )
 SELECT PART, LEVEL, SUBPART, QUANTITY
   FROM RPL;
```

이 조회는 예 1과 비슷합니다. 원래의 부품으로부터 레벨을 계산하기 위해 컬럼 *LEVEL*이 도입됩니다. 초기설정 fullselect에서 *LEVEL* 컬럼 값은 1로 초기설정됩니다. 후속 fullselect에서 상위 제품의 레벨은 1씩 증가합니다. 그런 후 결과의 레벨 수를 제어하기 위해 두 번째 fullselect에는 상위 레벨은 2 이하여야 한다는 조건이 포함됩니다. 이는 두 번째 fullselect는 하위 부품만이 두 번째 레벨이 되도록 합니다.

조회의 결과는 다음과 같습니다.

순환 예: BOM(Bill of Materials)

부품	레벨	부속 부품	수량
01	1	02	2
01	1	03	3
01	1	04	4
01	1	06	3
02	2	05	7
02	2	06	6
03	2	07	6
04	2	08	10
04	2	09	11
06	2	12	10
06	2	13	10

부록N. 예외 테이블

예외 테이블은 IMMEDIATE CHECKED 옵션을 가진 SET INTEGRITY를 사용하여 점검되도록 지정된 테이블의 정의를 모방하는 사용자 작성 테이블입니다. 이는 점검 중인 테이블의 제한조건을 위반하는 행의 사본을 저장하는 데 사용됩니다.

LOAD에 사용되는 예외 테이블은 여기에서 사용되는 것과 같습니다. 따라서 이는 SET INTEGRITY문으로 점검 중에 다시 사용할 수 있습니다.

예외 테이블 작성 규칙

예외 테이블을 작성하는 규칙은 다음과 같습니다.

1. 예외 테이블의 첫번째『n』개 컬럼은 점검 중인 테이블의 컬럼과 같습니다. 모든 컬럼 속성은 이름, 형태 및 길이를 포함하여 모두 같아야 합니다.
2. 예외 테이블의 모든 컬럼에는 제한조건과 트리거가 없어야 합니다. 제한조건에는 삽입시 오류를 발생시킬 수 있는 고유 색인 제한조건 뿐만 아니라 참조 무결성, 점검 제한조건이 포함됩니다.
3. 예외 테이블의『(n+1)』컬럼은 선택적인 TIMESTAMP 컬럼입니다. 이는 데이터 점검을 위해 SET INTEGRITY문을 발행하기 전에 예외 테이블내의 행이 삭제되지 않았다면, 같은 테이블에 대해 SET INTEGRITY문에 의한 점검의 후속 호출을 식별하는 데 사용됩니다.
4. 『(n+2)』컬럼은 CLOB(32K) 형태 또는 그 이상이어야 합니다. 이 컬럼은 선택적이나 사용하는 것이 좋습니다. 이 컬럼은 행 내의 데이터가 위반하는 제한조건의 이름을 부여하는 데 사용됩니다. 이 컬럼이 제공되지 않는 경우(예를 들어, 원래 테이블에 허용된 최대 컬럼 수가 있는 경우 보장됨), 제한조건 위반이 발견된 행들만 복사됩니다.
5. 예외 테이블은『(n+1)』과『(n+2)』컬럼으로 작성되어야 합니다.
6. 상기 추가 컬럼에 대한 특정한 이름을 강제하는 것은 없습니다. 그러나, 유형 스펙은 정확히 따라야 합니다.

7. 어떤 추가 컬럼도 허용되지 않습니다.
8. 원래 테이블에 DATALINK 컬럼이 있는 경우, 예외 테이블의 해당 컬럼은 NO LINK CONTROL을 지정해야 합니다. 이것은 (DATALINK 컬럼이 있는) 행이 삽입될 때 파일이 링크되지 않고, 예외 테이블에서 행들이 선택될 때 액세스 토큰이 생성되지 않도록 합니다.
9. 원래 테이블에 생성된 컬럼(IDENTITY 등록 정보 포함)이 있는 경우, 예외 테이블의 해당 컬럼은 생성된 등록 정보를 지정하지 말아야 합니다.
10. 데이터를 점검하기 위해 SET INTEGRITY를 호출하는 사용자는 예외 테이블에 대한 INSERT 특권을 가져야 함에 유의하십시오.

『message』 컬럼에 있는 정보는 다음 구조에 따른 것입니다.

표 145. 예외 테이블 메시지 컬럼 구조

필드 번호	목차	크기	주석
1	제한조건 위반 수	5자	'0'으로 채워 오른쪽 정렬
2	첫번째 제한조건 위반 유형	1자	'K' - 점검 제한조건 위반 'F' - 외부 키 위반 'G' - 생성된 컬럼 위반 'I' - 고유 색인 위반[a] 'L' - DATALINK 로드 위반
3	제한조건/컬럼[b]/색인 ID[c]/DLVDESC[d]의 길이	5문자	'0'으로 채워 오른쪽 정렬
4	제한조건 이름/컬럼 이름[b]/색인 ID[c]/ DLVDESC[d]	이전 필드로부터의 길이	
5	분리 기호	3자	<빈칸><콜론><빈칸>
6	다음 제한조건 위반의 유형	1문자	'K' - 점검 제한조건 위반 'F' - 외부 키 위반 'G' - 생성된 컬럼 위반 'I' - 고유 색인 위반 'L' - DATALINK 로드 위반
7	제한조건/컬럼/색인 ID/ DLVDESC의 길이	5문자	'0'으로 채워 오른쪽 정렬
8	제한조건 이름/컬럼 이름/색인 ID/ DLVDESC	이전 필드로부터의 길이	
.....	.....	.....	각 위반에 대해 필드 5-8 반복

표 145. 예외 테이블 메시지 컬럼 구조 (계속)

필드 번호	목차	크기	주석

- [a] SET INTEGRITY 사용 점검시 고유 색인 위반이 발생하지 않습니다. 그러나, FOR EXCEPTION 옵션을 선택하면 LOAD 실행시 보고됩니다. 즉 LOAD는 점검 제한조건, 생성된 컬럼 및 외부 키 위반을 예외 테이블에 보고하지 않습니다.
- [b] 카탈로그 뷰에서 생성된 컬럼의 표현식을 검색하려면 SELECT 문을 사용하십시오. 가령 예를 들어, 필드 4가 MYSCHEMA.MYTABLE.GEN_1인 경우, ″AS (<expression>)″ 양식에서 SELECT SUBSTR(TEXT, 1, 50) FROM SYSCAT.COLUMNS WHERE TABSCHEMA='MYSCHEMA' AND TABNAME='MYNAME' AND COLNAME='GEN_1'은 표현식의 처음 50문자를 리턴합니다.
- [c] 카탈로그 뷰에서 색인 ID를 검색하려면 SELECT 문을 사용하십시오. 예를 들어, 필드 4가 1234인 경우에는 SELECT INDSCHEMA, INDNAME FROM SYSCAT.INDEXES WHERE IID=1234입니다.
- [d]DLVDESC는 아래에 설명된 DATALINK 로드 위반 DESCriptor입니다.

표 146. DATALINK 로드 위반 DESCriptor (DLVDESC)

필드 번호	목차	크기	주석
1	DATALINK 위반 컬럼 수	4자	'0'으로 채워 오른쪽 정렬
2	첫번째 위반 컬럼의 DATALINK 컬럼 수	4자	'0'으로 채워 오른쪽 정렬
2	두 번째 위반 컬럼의 DATALINK 컬럼 수	4자	'0'으로 채워 오른쪽 정렬
.....	.....	.....	위반 컬럼 번호 각각에 대한 반복

주:

- DATALINK 컬럼 번호는 해당 테이블에 대한 SYSCAT.COLUMNS의 COLNO입니다.

예외 테이블의 행 처리

예외 테이블의 정보는 원하는 대로 처리될 수 있습니다. 행을 사용하여 데이터를 정정하고 그 행을 원래의 테이블로 재삽입할 수도 있습니다.

원래 테이블에 INSERT 트리거가 없는 경우, 예외 테이블에 있는 부속 조회로 INSERT문을 발행함으로써 정정된 행을 전송하십시오.

INSERT 트리거가 있고 트리거를 시작하지 않고 예외 테이블의 정정된 행으로 적재를 완료하려는 경우, 다음 방법을 사용하는 것이 좋습니다.

- 목적에 맞도록 정의된 컬럼의 값에 따라 시작될 INSERT 트리거를 설계합니다.
- 예외 테이블의 데이터를 적재 해제하고 LOAD를 사용하여 이를 추가합니다. 이 때 데이터를 재점검한다면, DB2 버전 7에서 제한조건 위반 점검이 추가된 행에만 한정되지 않는다는 점을 명기해야 합니다.
- 관련 카탈로그 테이블의 트리거 원문을 저장합니다. 그런 후, INSERT 트리거를 삭제하고 INSERT를 사용하여 예외 테이블의 정정된 행을 전송합니다. 마지막으로, 저장된 정보를 사용하여 트리거를 재작성합니다.

DB2 버전 7에서, 예외 테이블의 행을 삽입할 경우 트리거를 제거할 수 없도록 금지하는 명시적 규정은 없습니다.

고유 색인 위반에 대해 행당 하나의 위반만이 보고됩니다.

LONG 문자열 또는 LOB 데이터 유형을 가진 값이 테이블에 있는 경우, 고유 색인 위반의 경우 값은 예외 테이블에 삽입되지 않습니다.

예외 테이블 조회

예외 테이블의 메시지 컬럼 구조는 이전에 기술한 바와 같이, 제한조건 이름, 길이 및 분리 문자의 병합된 목록입니다. 이 정보에 관하여 조회를 기록 할 수 있습니다.

예를 들어 제한조건 이름만 있는 각 행을 반복하여 모든 위반 목록을 얻기 위한 조회를 작성합시다. 원래 테이블 T1에는 C1과 C2의 두컬럼이 있다고 가정합시다. 또한, 해당 예외 테이블 E1은 메시지 컬럼으로서 컬럼 C1을 가지고, T1과 MSGCOL에 있는 것과 관계된 C2를 가진다고 가정합시다. 다음 조회(순환 사용)는 행당 하나의 제한조건 이름을 나열합니다(둘 이상의 위반에 대해서는 행 반복).

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
  (SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, 12,
                   INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
       1,
       15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
     FROM E1
   UNION ALL
   SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, J+6,
                 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
       I+1,
```

```
      J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
    FROM IV
    WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
) SELECT C1, C2, CONSTNAME FROM IV;
```

특정 제한조건을 위반한 모든 행을 원할 경우, 이 조회를 다음과 같이 확장할 수 있습니다.

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
  (SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, 12,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
      1,
      15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
    FROM E1
  UNION ALL
  SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, J+6,
                INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
      I+1,
      J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
    FROM IV
    WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTNAME = 'constraintname';
```

모든 점검 제한조건 위반을 얻기 위해 다음을 실행할 수 있습니다.

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, CONSTTYPE, I, J) AS
  (SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, 12,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
      CHAR(SUBSTR(MSGCOL, 6, 1)),
      1,
      15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
    FROM E1
  UNION ALL
  SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, J+6,
                INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
      CHAR(SUBSTR(MSGCOL, J, 1)),
      I+1,
      J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
    FROM IV
    WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTTYPE = 'K';
```

부록O. 일본어 및 대만어의 EUC 고려사항

일본어 및 대만어에 대한 확장 Unix 코드(EUC: Extended Unix Code)는 1개에서 4개까지의 문자 세트를 지원할 수 있는 코드화 규칙 세트를 정의합니다. 일본어 EUC(eucJP) 및 대만어 EUC(eucTW)와 같은 경우, 세 바이트 이상을 사용하여 한 문자를 코드화할 수 있습니다. 이러한 코드화 체계의 사용에는 데이터베이스 서버 또는 데이터베이스 클라이언트의 코드 페이지로 사용되는 경우가 포함됩니다. 키 고려사항에는 다음 내용이 포함됩니다.

- EUC 코드 페이지 및 2바이트 코드 페이지 사이를 변환할 때 문자열의 확장 또는 축소.
- eucJP(일본어) 또는 eucTW(대만어) 코드 페이지로 정의된 데이터베이스 서버에 저장된 그래픽 데이터에 대한 코드 페이지로서 UCS-2(범용 문자 세트-2)를 사용.

이들 고려사항에는 예외도 있으나, EUC를 사용하는 것은 2바이트 문자 세트(DBCS) 지원에 일치합니다. 이 책에서 *2바이트*로 언급한 것은 3바이트 이상을 필요로 하는 문자 표시를 위해 허용하는 코드화 규칙 지원을 반영하도록 *복수 바이트*로 변경되었습니다. 일본어와 대만어 EUC를 지원하기 위한 자세한 고려사항이 여기에 포함되어 있고, 이 책의 목차와 같은 방식으로 편성되어 있습니다. EUC 데이터베이스 서버 또는 EUC 데이터베이스 클라이언트에서 SQL을 사용할 경우, *응용프로그램 개발 안내서*의 응용프로그램 개발 환경과 연계하여 이 정보를 사용해야 합니다.

언어 요소

문자

각 복수 바이트 문자는 *특수 문자*로 간주되는 2바이트 공백 문자를 예외로 하고 *글자*로 간주됩니다.

토큰

복수 바이트 소문자는 대문자로 변환되지 않습니다. 이는 일반적으로 대문자로 변환되는 토큰에 있는 1바이트 소문자와는 다릅니다.

식별자

SQL 식별자

2바이트 코드 페이지와 EUC 코드 페이지 사이의 변환 결과 2바이트 문자가 2바이트 이상으로 코드화된 복수 바이트 문자로 변환될 수 있습니다. 따라서, 2바이트 코드 페이지의 최대 길이에 맞는 식별자는 EUC 코드 페이지의 길이를 초과할 수 있습니다. 최대 식별자 길이를 넘지 않도록 하기 위해서는 이 유형의 환경에 대한 식별자를 주의깊게 선택해야 합니다.

데이터 유형

문자열

MBCS 데이터베이스에서는, 문자열에 1바이트 문자 세트(SBCS)와 다중 바이트 문자 세트(MBCS)의 문자가 혼합될 수도 있습니다. 그러한 문자열을 사용할 때, 문자에 기초했거나(데이터를 문자로 다룸) 바이트에 기초한 경우(데이터를 바이트로 다룸), 연산 결과 다른 결과가 나올 수도 있습니다. 혼합 문자열이 처리되는 방식을 결정하려면 연산(조작) 또는 함수 설명을 점검하십시오.

그래픽 문자열

그래픽 문자열은 일련의 2바이트 문자 데이터로 정의됩니다. 일본어나 대만어 EUC 데이터가 그래픽 컬럼에 보관될 수 있도록, EUC 문자는 UCS-2로 코드화됩니다. 지원되는 모든 코드화 체계(PC 또는 EBCDIC DBCS)에서 2바이트 문자가 아닌 문자는 그래픽 컬럼에 사용할 수 없습니다. 2바이트 문자가 아닌 것을 사용할 경우, 변환 중에 대체 문자로 교체될 수 있습니다. 그러한 데이터를 검색할 경우, 입력한 것과 같은 값을 리턴하지 않습니다. 호스트 변수에서 그래픽 데이터를 처리하는 데 대한 세부사항은 *응용프로그램 개발 안내서* 프로그래밍 언어 절을 참조하십시오.

지정 및 비교

문자열 지정

문자열 변환은 지정 이전에 수행됩니다. eucJP/eucTW 코드 페이지와 DBCS 코드 페이지를 포함시키는 경우, 문자열은 더 길어지거나(DBCS에서 eucJP/eucTW로) 더 짧아집니다(eucJP/eucTW에서 DBCS로). 이럴 경우, 검색 지정시 절단이 생기고 저장영역 지정에 오류가 발생할 수 있습니다. 저장영역 지정의 오류가 변환중의 확대에 기인한 것일 경우, SQLSTATE 22524가 SQLSTATE 22001 대신 리턴됩니다.

마찬가지로, 그래픽 문자열을 포함하여 지정할 경우, UCS-2 코드화 2바이트 문자는 해당 2바이트 문자가 없는 문자에 대해 PC 또는 EBCDIC DBCS 코드 페이지에 있는 대체 문자로 변환될 수 있습니다. 대체 문자로 문자를 교체하는 지정은 SQLCA의 SQLWARN10 필드를 'W'로 설정함으로써 이를 표시합니다.

다중 바이트 문자열이 수반된 검색 지정 동안 절단이 발생하는 경우에는, 절단점이 다중 바이트 문자의 부분이 될 수도 있습니다. 이 경우에는, 문자 단편의 각 바이트가 1바이트의 공백으로 대체됩니다. 이는 두 개 이상의 1바이트 공백이 절단된 문자열 끝에 나타날 수도 있음을 의미합니다.

문자열 비교

문자열 비교는 바이트 단위로 이루어집니다. 문자열은 데이터베이스에 대해 정의된 조합 순서를 사용합니다. 그래픽 문자열은 조합 순서를 사용하지 않으며, eucJP 또는 eucTW 데이터베이스에서는 UCS-2를 사용하여 코드화됩니다. 따라서, 두 개의 혼합 문자열의 비교 결과는, 여기에 같은 문자가 들어 있더라도 두 개의 그래픽 문자열의 비교 결과와 다를 수 있습니다. 마찬가지로, 혼합 문자 컬럼 및 그래픽 컬럼의 정렬 순서는 다를 수 있습니다.

결과 데이터 유형 규칙

문자열에 대한 결과 데이터 유형은 문자열이 확대될 경우에도 영향받지 않습니다. 예를 들어, 두 개의 CHAR 피연산자가 결합되어도 여전히 CHAR이 됩니다. 그러나, 최대 확대를 함으로써 길이 속성이 두 피연산자 중 더 큰 것으로 되는 것과 같이, 한 문자열 피연산자가 변환되는 경우 그 문자열 길이 속성은 영향받습니다. 예를 들어, VARCHAR(100) 및 VARCHAR(120)의 데이터 유형을 가지는 CASE

표현식의 결과 표현식을 고려하십시오. VARCHAR(100) 표현식이 (변환이 필요한) 혼합 문자열 호스트 변수이고 VARCHAR(120) 표현식은 eucJP 데이터베이스의 컬럼이라고 가정합니다. 그 결과 데이터 유형은 VARCHAR(200)이 됩니다. VARCHAR(100)이 가능한 변환을 위해 두 배로 되었기 때문입니다. eucJP 또는 eucTW 데이터베이스가 관련되지 않은 동일한 시나리오는 VARCHAR(120)의 결과 유형을 갖게 됩니다.

호스트 변수 길이를 두배로 하는 것은 데이터베이스 서버가 일본어 EUC나 대만어 EUC라는 사실에 근거한다는 것을 유념하십시오. 클라이언트가 eucJP 또는 eucTW인 경우에도, 여전히 두배로 됩니다. 이로써 2바이트 또는 복수 바이트 클라이언트에 의해 같은 응용프로그램 패키지가 사용됩니다.

문자열 변환 규칙

SQL 참조서의 해당 절에 나열된 연산의 유형은 피연산자를 응용프로그램이나 데이터베이스 코드 페이지로 변환시킵니다.

그러한 조작이 일본어 EUC와 대만어 EUC를 포함하는 혼합된 코드 페이지 환경에서 행해지면, 혼합된 문자열 피연산자의 확장이나 축약이 발생할 수도 있습니다. 그러므로, 결과 데이터 유형에는 가능한 경우 최대 확대 크기를 수용할 수 있는 길이 속성이 포함됩니다. 데이터 유형의 길이 속성에 대한 제한이 있다면, 데이터 유형에 허용되는 최대 길이가 사용됩니다. 예를 들어, 최대 성장이 두배인 환경에서, VARCHAR(200) 호스트 변수는 VARCHAR(400)인 것처럼 취급되나, CHAR(200) 호스트 변수는 CHAR(254)인 것처럼 취급됩니다. 변환된 문자열이 데이터 유형의 최대 길이를 초과하는 경우, 변환 런타임 오류가 발생할 수 있습니다. 예를 들어, CHAR(200)과 CHAR(10)을 결합하면 CHAR(254)라는 결과 유형이 생깁니다. UNION의 왼쪽 값이 변환될 때, 254자 이상이 필요한 경우 오류가 발생합니다.

어떤 경우, 변환에 최대 증대 길이를 허용하면 길이 속성이 한계를 넘게 됩니다. 예를 들어, UNION은 최고 254자만을 허용합니다. 즉, 가변 길이 문자열 128바이트 길이로 정의된 DBCS 혼합 문자열인 컬럼 목록(hv1)에 호스트 변수를 포함시킨 union이 있는 조회는 데이터 유형을 VARCHAR(256)로 설정하며, 그 결과

응용프로그램의 조회가 254를 넘는 컬럼에 표시되지 않는 경우에도 조회 준비 오류가 발생합니다. 실제 문자열이 254바이트를 넘지 않는 상황에서 다음을 사용하여 명령문을 준비할 수 있습니다.

```
SELECT CAST(:hv1 CONCAT ' AS VARCHAR(254)), C2 FROM T1
UNION
SELECT C1, C2 FROM T2
```

호스트 변수와 널(NULL) 문자열의 병합은 유형변환이 수행되기 전에 강제로 변환이 발생됩니다. 이 조회는 런타임 절단 오류가 발생하는 경우에도 DBCS에서 eucJP/eucTW 환경으로 준비될 수 있습니다.

이 기술(유형변환으로 널(null) 문자열 병합)은 SELECT DISTINCT에 대한 유사한 254바이트 한계를 처리하거나 ORDER BY 또는 GROUP BY절에서 컬럼 사용시에 사용됩니다.

상수

그래픽 리터럴

그래픽 리터럴에는 단일 또는 다중 바이트 문자(혼합 문자열과 유사)이 포함될 수 있습니다. 문자열에는 2 000자 이상이 포함될 수 없습니다. 모든 관련 PC 및 EBCDIC 2바이트 코드 페이지에서 2바이트 문자로 변환되는 문자만을 그래픽 상수로 사용하는 것이 좋습니다. SQL문의 그래픽 리터럴은 클라이언트 코드 페이지로부터 데이터베이스 서버의 2바이트 코드로 변환됩니다. 일본어 또는 대만어 EUC 서버의 경우, 상수는 그래픽 문자열에 사용되는 2바이트 코드인 UCS-2로 변환됩니다. 2바이트 서버의 경우, 상수는 클라이언트 코드 페이지에서 서버의 DBCS 코드 페이지로 변환됩니다.

함수

사용자 정의 함수를 고안할 때에는 매개변수 데이터 유형에 대한 일본어 또는 대만어 EUC 지원에 따른 영향을 고려해야 합니다. 함수 차수의 경우 함수 호출에 대한 인수의 데이터 유형을 고려합니다. 일본어 또는 중국어 EUC 클라이언트를 포함하는 혼합 문자열 인수에는 인수 지정에 필요한 추가 바이트가 필수적입니다. 이는 데이터 유형이 길이 증가를 허용하여 변경되도록 합니다. 예를 들어, 서버에 있는 VARCHAR(4000) 문자열에 맞는 응용프로그램의 문자열(LONG

VARCHAR)을 나타내려면 4001바이트가 필요합니다. 함수 시그니처가 포함되어 있지 않아 인수가 LONG VARCHAR이 된 경우, 함수 차수는 함수를 찾는 데 실패합니다.

다양한 이유로 LONG 문자열을 허용하지 않는 함수도 있습니다. 그러한 함수와 함께 LONG VARCHAR 또는 CLOB 인수를 사용하면 성공하지 못합니다. 예를 들어, 내장 POSSTR 함수의 두 번째 인수인 LONG VARCHAR가 함수 차수에 실패합니다(SQLSTATE 42884).

표현식

병합 연산자가 있는 표현식

병합 피연산자 중 하나를 확대시키면, 일본어 또는 대만어 EUC 데이터베이스 서버가 포함된 환경에서, 데이터 유형 및 병합 피연산자의 길이가 변경될 수 있습니다. 예를 들어, 호스트 변수 값의 길이가 두배로 되는 EUC 서버의 경우, 다음 예를 생각해 보십시오.

```
CHAR200 CONCAT :char50
```

컬럼 *CHAR200*은 유형 CHAR(200)입니다. 호스트 변수 *char50*은 CHAR(50)으로 정의됩니다. 이 병합 연산의 결과 유형은 대개 CHAR(250)입니다. 그러나, eucJP 또는 eucTW 데이터베이스 서버를 고려할 때, 문자열 길이가 두 배로 될 수 있다고 추측할 수 있습니다. 그러므로, *char50*은 CHAR(100)로 취급되고 그 결과 데이터 유형은 VARCHAR(300)입니다. 결과가 VARCHAR인 경우에도, 후미 공백을 포함하여 항상 300바이트의 데이터를 가짐에 유의하십시오. 추가로 후미 공백이 생기지 않도록 하려면, 호스트 변수를 CHAR(50) 대신 VARCHAR(50)으로 정의하십시오.

술어

LIKE 술어

EUC 데이터베이스의 혼합 문자열을 포함하는 LIKE 술어의 경우,

- 1바이트 밑줄은 1개 1바이트 문자를 나타냅니다.
- 1바이트 퍼센트는 제로 또는 그 이상의 문자열(1바이트 또는 복수 바이트 문자)을 나타냅니다.

- 2바이트 밑줄은 1개의 복수 바이트 문자를 나타냅니다.
- 2바이트 퍼센트는 제로 또는 그 이상의 문자열(1바이트 또는 복수 바이트 문자)을 나타냅니다.

escape 문자는 1바이트 문자 1개 또는 2바이트 문자 1개여야 합니다.

밑줄 문자 사용시, LIKE 조작의 코드 페이지에 따라 그 결과가 달라질 수 있습니다. 예를 들어, 일본어 EUC로 된 카타카나 문자는 복수 바이트 문자(CS2)나, 일본어 DBCS 코드 페이지로될 경우 1바이트 문자입니다. *패턴 표현식*에 있는 단일 바이트 밑줄을 가진 조회는 일본어 DBCS 서버로부터 밑줄 위치에 있는 카타카나 문자 발생을 리턴합니다. 그러나, 일본어 EUC 서버로 된 해당 테이블의 동일한 행은 리턴되지 않을 것입니다. 카타카나 문자는 2바이트 밑줄과만 일치하기 때문입니다.

EUC 데이터베이스의 그래픽 문자열이 포함된 LIKE 술어의 경우,

- 밑줄 및 퍼센트에 사용된 문자는 각각 밑줄 및 퍼센트 문자로 대응되어야 합니다.
- 밑줄은 1개의 UCS-2문자를 나타냅니다.
- 퍼센트는 제로 또는 그 이상의 UCS-2 문자열을 나타냅니다.

함수

LENGTH

이 함수의 처리는 EUC 환경에 있는 혼합 문자열의 경우 다릅니다. 리턴된 값은 인수의 코드 페이지에 있는 문자열의 길이입니다. 이 함수를 사용하여 값의 길이를 결정할 때, 길이를 어떻게 사용할 것인지에 대해 신중하게 고려해야 합니다. 이는 특히 혼합 리터럴의 경우일 때 해당되는데, 길이가 문자가 아닌 바이트 단위로 주어지기 때문입니다. 예를 들어, LENGTH 함수에 의해 리턴되는 DBCS 데이터베이스에 있는 혼합 문자열 컬럼의 길이가, 일부 DBCS 문자는 복수 바이트 eucJP 또는 eucTW 문자로 변환되므로, eucJP 또는 eucTW 문자에 있는 해당 컬럼의 검색된 값이 길이보다 작을 수 있습니다.

SUBSTR

SUBSTR 함수는 바이트에 기초하여 혼합 문자열에 대해서 작동됩니다. 따라서, 결과로 산출되는 문자열에는 문자열의 시작 또는 끝부분에 복수 바이트 문자의 단편들이 포함될 수 있습니다. 문자 단편의 검출 또는 처리가 제공되지 않습니다.

TRANSLATE

TRANSLATE 함수는 다중 바이트 문자열을 비롯하여 혼합 문자열을 지원합니다. *to-string-exp* 및 *from-string-exp*의 해당 문자는 동일한 바이트 수를 가져야 하며 다중 바이트 문자의 일부로 끝날 수 없습니다.

*채움 문자 표현식*은 *문자열 표현식*이 문자열일 때 1바이트 문자가 되어야 합니다. TRANSLATE가 *char-문자열 표현식*의 코드 페이지에서 수행되므로, *채움 문자열 표현식*은 복수 바이트 문자에서 1바이트 문자로 변환될 수 있습니다.

복수 바이트 문자의 일부로 끝나는 *문자열 표현식*에는 변환된 바이트가 없습니다.

VARGRAPHIC

일본어나 대만어 EUC 코드 페이지로 된 문자열 피연산자에 대한 VARGRAPHIC 함수는 UCS-2 코드 페이지로 그래픽 문자열을 리턴합니다.

- 1바이트 문자는 우선 자신이 속한 코드 집합(eucJP 또는 eucTW)에 있는 해당 2바이트 문자로 변환됩니다. 그런 후, 해당 UCS-2 표현으로 변환됩니다. 2바이트 표현이 없는 경우, 문자는 UCS-2 표현으로 변환되기 전에 그 코드 집합에 대해 정의된 2바이트 대체 문자로 변환됩니다.
- 카타카나(eucJP CS2)인 eucJP로부터의 문자가 일부 코드화 체계로 된 1바이트 문자입니다. 따라서, 이들은 eucJP로 된 2바이트 문자로 변환되거나, UCS-2로 변환되기 전에 2바이트 대체 문자로 변환됩니다.
- 복수 바이트 문자는 UCS-2 표현으로 변환됩니다.

명령문

CONNECT

CONNECT문을 연속하여 처리하면, 클라이언트나 서버에 일본어 또는 대만어 EUC 코드 페이지를 포함하는 환경에서 응용프로그램이 데이터를 처리할 가능성이 있을 경우 중요한 정보가 SQLCA로 리턴됩니다. *SQLERRD(1)* 필드는 응용프로그램 코드 페이지에서 데이터베이스 코드 페이지로 변환될 때 혼합 문자열의 최대 확장을 제시합니다. *SQLERRD(2)* 필드는 데이터베이스 코드 페이지로부터 응용프로그램 코드 페이지로 변환될 때 혼합 문자열이 최대로 확장됩니다. 축약이 발생할 수 있는 경우에는 음수로, 확장이 발생할 수 있는 경우에는 양수가 됩니다. 음수값인 경우, 최악의 경우는 축약이 발생하지 않고 변환 후에 문자열의 전체 길이가 필요한 경우이므로, 그 값은 항상 -1입니다. 양수값은 2 정도가 되며, 이는 최악의 경우 변환 후 문자열에 필요한 문자열 길이가 2배로 됨을 의미합니다.

또한, 응용프로그램서버(AS) 및 응용프로그램 클라이언트의 코드 페이지는 SQLCA의 SQLERRMC 필드에서도 사용할 수 있습니다.

PREPARE

(1090 페이지의 표28에 기술된) 미입력 매개변수 표시문자에 대한 데이터 유형은 일본어나 대만어 EUC가 포함된 환경에서는 변경되지 않습니다. 그 결과, 입력 매개변수 표시문자를 사용하여 eucJP 또는 eucTW에 혼합 문자열에 대한 충분한 길이를 제공하는 것이 필요할 경우도 있습니다. 예를 들어, CHAR(19) 컬럼으로의 삽입을 고려하십시오. 다음 명령문을 준비하는 경우,

```
INSERT INTO T1 (CH10) VALUES (?)
```

매개변수 표시문자에 대해 데이터 유형은 CHAR(10)이 됩니다. 클라이언트가 eucJP 또는 eucTW인 경우, 삽입될 문자열을 표시하는 데 10바이트 이상이 필요하나, 데이터베이스의 DBCS 코드 페이지에 있는 같은 문자열은 10바이트 이하입니다. 이런 경우, 준비 명령문에는 10보다 큰 길이를 가진 입력 매개변수 표시문자가 포함되어야 합니다. 그러므로, 다음 명령문을 준비하면

```
INSERT INTO T1 (CH10) VALUES (CAST(? AS VARCHAR(20))
```

매개변수 표시문자에 대해 데이터 유형은 VARCHAR(20)이 됩니다.

부록P. DATALINK를 위한 BNF 스펙

DATALINK 값은 데이터베이스로부터 데이터베이스 외부에 저장된 파일에 이르기까지 논리 참조가 포함된 캡슐화된 값입니다.

캡슐화된 이 값의 자료 위치 속성은 URL(Uniform Resource Locator) 양식인 파일로의 논리적 참조입니다. 이 속성의 값은 RFC 1738을 기본으로 다음 BNF에 의해 지정된 대로[119] URL의 구문을 따릅니다: Uniform Resource Locators(URL), T. Berners-Lee, L. Masinter, M. McCahill, December 1994

다음 규약이 BNF 스펙에서 사용됩니다.

- "|"는 대체를 지정하는 데 사용됩니다.
- 브라켓[ ]은 선택적 또는 반복되는 요소 주위에 사용됩니다.
- 리터럴은 ""으로 묶입니다.
- 요소 앞에는 [*n*]*이 올 수 있는데, 이는 다음 요소의 *n*번 이상 반복을 나타냅니다. *n*이 지정되지 않을 경우 기본값은 0입니다.

DATALINK를 위한 BNF 스펙은 다음과 같습니다.

URL

```
url          =    httpurl | fileurl | uncurl | dfsurl | emptyurl
```

HTTP

```
httpurl      =    "http://" hostport [ "/" hpath ]
hpath        =    hsegment *[ "/" hsegment ]
hsegment     =    *[ uchar | ";" | ":" | "@" | "&" | "=" ]
```

RFC1738의 원래 BNF에서 검색 요소가 제거되었는데, 이는 검색 요소가 파일 참조의 중요한 부분이 아니고 DATALINK 문맥에서 아무 의미도 없기 때문입니다.

FILE

119. BNF는 제공된 언어의 구문을 설명하는 공식 표기법인 "Backus Naur Form"의 약어입니다.

```
fileurl      =    "file://" host "/" fpath
fpath        =    fsegment *[ "/" fsegment ]
fsegment     =    *[ uchar | "?" | ":" | "@" | "&" | "=" ]
```

RFC1738과는 대조적으로, host는 선택적이 아니며 "localhost" 문자열은 특별한 의미를 지니지 않습니다. 이는 클라이언트/서버 및 EEE 구성에서 "localhost"의 해석이 혼동되는 것을 막습니다.

UNC

```
uncurl       =    "unc:\\" hostname "\" sharename "\" uncpath
sharename    =    *uchar
uncpath      =    fsegment *[ "\" fsegment ]
```

NT에서 일반적으로 사용되는 UNC 이름 지정 규칙을 지원합니다. 이것은 RFC1738의 표준 스킴이 아닙니다.

DFS

```
dfsurl        =   "dfs://.../" cellname "/" fpath
cellname      =   hostname
```

DFS 이름 지정 스킴을 지원합니다. 이것은 RFC1738의 표준 스킴이 아닙니다.

EMPTYURL

```
emptyurl     =    ""
hostport     =    host [ ":" port ]
host         =    hostname | hostnumber
hostname     =    *[ domainlabel "." ] toplabel
domainlabel  =    alphadigit | alphadigit *[ alphadigit | "-" ] alphadigit
toplabel     =    alpha | alpha *[ alphadigit | "-" ] alphadigit
alphadigit   =    alpha | digit
hostnumber   =    digits "." digits "." digits "." digits
port         =    digits
```

DATALINK 값에 대해 빈(길이가 0) URL도 지원됩니다. 이는 조정 예외가 보고되고 널(NULL)이 아닌 DATALINK 컬럼이 호출될 때 DATALINK 컬럼을 갱신하는 데 사용됩니다. 길이가 0인 URL은 컬럼을 갱신하고 링크를 해제시키는 데 사용됩니다.

기타 정의

```
lowalpha      =   "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
                  "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
                  "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
```

```
                    "y" | "z"
hialpha         =   "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
                    "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" |
                    "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
                    "Y" | "Z"
alpha           =   lowalpha | hialpha
digit           =   "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                    "8" | "9"
safe            =   "$" | "-" | "_" | "." | "+"
extra           =   "!" | "*" | "'" | "(" | ")" | ","
hex             =   digit | "A" | "B" | "C" | "D" | "E" | "F" |
                    "a" | "b" | "c" | "d" | "e" | "f"
escape          =   "%" hex hex
unreserved      =   alpha | digit | safe | extra
uchar           =   unreserved | escape
digits          =   1*digit
```

앞 및 뒤 공백 문자는 분석 중에 DB2에 의해 제거됩니다. 또한 스킴 이름('HTTP', 'FILE', 'UNC', 'DFS') 및 host는 대소문자가 구별되며 데이터베이스에 항상 대문자로 저장됩니다.

부록Q. 용어집

가

가변 길이 문자열. 그 길이가 고정되어 있지 않으나 설정 한계를 범위로 하는 문자, 그래픽 또는 2진 문자열. *가변 길이 문자열*이라고도 합니다.

가변 함수. 그 결과가 다른 요소뿐만 아니라 입력 매개변수 값에 종속되는 사용자 정의 함수. 동일한 매개변수 값으로 연속 호출하면 서로 다른 결과가 산출될 수 있습니다. 반의어 - *불변 함수*.

가상(phantom) 행. 반복 읽기(RR)를 제외한 분리 레벨에서 실행중인 응용프로그램 프로세스가 읽을 수 있는 테이블 행. 응용프로그램 프로세스가 단일 작업 단위 내에서 여러 번 동일한 조회를 발행하는 경우, 동시에 수행중인 응용프로그램 프로세스에 의해 데이터가 삽입 및 확약되므로 조회 사이에 추가 행이 나타날 수 있습니다.

가져오기. PC/IXF, DEL, WSF 또는 ASC와 같은 형식을 사용하여 외부 파일로부터 데이터를 데이터베이스 관리 프로그램 테이블로 복사하는 것. 반의어 - *내보내기*.

가져오기 유틸리티. 사용자 제공 레코드 데이터를 테이블로 삽입하는 트랜잭션 유틸리티. 반의어 - *로드 유틸리티*.

간격 타이밍. DB2 복제에서, 복사 작업 내역 순환을 시작하는 시기를 제어하는 가장 간단한 방법. 복사 작업 내역 순환의 시작 날짜와 시간을 지정하고, 복사 작업 내역 순환을 수행할 빈도를 설명하는 시간 간격을 설정해야 합니다. 반의어 - **이벤트 타이밍** 및 **요구 시 타이밍**.

강화된 충돌 검출. 모든 Replica 및 소스 테이블에서의 데이터 무결성을 보장하는 충돌 검출. Apply 프로그램은 차후 트랜잭션에 대해 복사 작업 내역 세트의 사용자 테이블이나 모든 Replica를 잠급니다. 잠그기 전에 수행된 변경사항이 캡쳐된 후에 검출을 시작합니다. *충돌 검출*, *표준 충돌 검출* 및 *행 Replica 충돌 검출*도 참조.

개인 연결. OS/390용 DB2 UDB에 고유한 통신 연결.

개인 프로토콜 액세스. 조회를 또 다른 DB2 시스템에 보낼 수 있는 분산 분산 데이터 액세스 방법. 반의어 - *DRDA 액세스*.

개인 프로토콜 연결. 응용프로그램 프로세스의 DB2 개인 연결. *개인 연결*도 참조.

갭. DB2 복제에서, Capture 프로그램이 로그 또는 저널 레코드를 읽을 수 없어서 변경 데이터가 유실될 가능성이 있는 상태.

갱신 규칙. 컬럼을 갱신하기 위해 충족시켜야 하는 데이터베이스 관리 프로그램 조건.

갱신 트리거. OS/390용 DB2 UDB에서, 트리거링 SQL 조작 UPDATE에서 정의된 트리거.

거부 트랜잭션. DB2 복제에서, 소스 테이블에 대한 비교에서 날짜가 벗어나는 복제 테이블의 여러 갱신사항을 포함하는 트랜잭션.

거짓 전역 잠금 경합. OS/390용 DB2 UDB에서, 여러 잠금 이름이 동일한 표시기로 해싱되고 어떤 실제 경합도 존재하지 않을 경우에 결합 기능으로부터 경합이 표시되는 것.

검색 조건. 테이블에서의 행 선택 기준. 검색 조건은 하나 이상의 술어로 구성됩니다.

결과 세트 위치 지정자. 저장 프로시듀어가 리턴하는 조회 결과 세트를 고유하게 식별하기 위해 OS/390용 DB2 UDB에서 사용하는 4 바이트의 값.

결과 집합. 저장 프로시듀어가 리턴하는 행들의 세트.

결과 테이블. SELECT문의 평가에 의해 산출되는 행 집합.

결정 함수. *불변 함수* 참조.

결합 기능. OS/390 환경에서, 결합 기능 제어 프로그램을 실행하거나 고속 캐싱, 목록 처리 및 병렬 Sysplex에서의 함수 잠금을 제공하는 특수 PR/SM™ LPAR 논리적 파티션.

경고. 성능 변수가 경고나 경보 역치 아래이거나 이를 초과할 때 생성되는 경보음이나 경고와 같은 조치.

경로. *SQL 경로* 참조.

경합. 데이터베이스 관리 프로그램에서, 트랜잭션이 이미 잠긴 테이블 또는 행을 잠그려고 하는 상황.

계승. 클래스 계층의 상위 클래스 다운스트림에서 하위 클래스로 클래스 자원 또는 속성을 전달하는 것.

계정 문자열. DB2 Connect에 의해 DRDA®에 보내진 사용자 정의 계정 정보. 이 정보는 다음 위치 중 하나에서 지정될 수 있습니다.

- SQLESACT API 또는 DB2ACCOUNT 환경 변수를 사용하는 클라이언트 워크스테이션
- DFT_ACCOUNT_STR 데이터베이스 관리 프로그램 구성 매개변수를 사용하는 DB2 Connect 워크스테이션.

고유 색인. 동일한 키 값이 한 테이블에 저장될 수 없도록 하는 색인.

고유 제한조건. 기본 키 또는 고유 색인 키에 있는 두 값은 같아서는 안 되는 규칙. *고유성 제한조건*이라고도 합니다.

고유 키. 두 값이 동일한 값을 갖지 않도록 제한받는 키.

고정 길이 문자열. 길이가 지정되어 있고 변경할 수 없는 그래픽 문자열 또는 문자열. 반의어 - *가변 길이 문자열*.

공용 권한. 모든 사용자에게 부여된 오브젝트에 대한 권한 부여.

공유 잠금. 동시에 실행 중인 응용프로그램 프로세스를 데이터베이스 데이터에 대한 읽기 전용 연산으로 제한하는 잠금. 반의어 - *독점 잠금*.

공유된 통신 영역(SCA). OS/390용 DB2 UDB 데이터 공유 그룹에서 DB2간 통신을 위해 사용하는 결합 기능 목록 구조.

공통 색인 테이블. 텍스트 컬럼이 공통 텍스트 색인을 공유하는 DB2 테이블. 또한 *복수 색인 테이블*도 참조.

공통 서비스 영역(CSA). OS/390에서, 모든 주소 공간에서 주소지정될 수 있는 데이터 영역을 포함하는 공통 영역의 부분.

공통 테이블 표현식. WITH절 뒤에 오는 fullselect의 FROM절에 있는 테이블 이름으로서 지정될 수 있는 이름(규정된 SQL 식별자)으로 결과 테이블을 정의하는 표현식.

공통 프로그래밍 인터페이스 통신(CPI-C). 프로그램간 통신을 요구하는 응용프로그램에 대한 API로, SNA LU 6.2를 사용하여 프로그램간 서비스 세트를 작성합니다.

관계. OS/390용 DB2 UDB에서, 한 테이블 행들이나 두 테이블의 행들 사이의 정의된 연결. 관계는 참조 제한조건의 내부 표시입니다.

관계 큐브. 함께 다차원 데이터베이스를 정의하는 데이터 및 메타데이터 세트. 관계형 큐브는 관계형 데이터베이스에 저장되는 다차원 데이터베이스의 부분입니다. 또한 *다중 차원 데이터베이스*.

관계형 데이터베이스. 데이터의 관계형 모델에 따라 조작되고 테이블 세트로서 인식될 수 있는 데이터베이스.

관계형 데이터베이스 관리 시스템(RDBMS). OS/390용 DB2 UDB에서, 관계형 데이터베이스에 대한 액세스를 구성하고 제공하는 하드웨어 및 소프트웨어 콜렉션.

관계형 데이터베이스 이름(RDBNAM). 네트워크내의 RDBMS에 대한 고유 식별자. OS/390용 DB2 UDB에서, 이것은 CDB에 있는 SYSIBM.LOCATIONS 테이블의 LOCATION 컬럼 값이어야 합니다. OS/390용 DB2 UDB 서적에서는 다른 RDBMS 이름을 LOCATION 값이나 위치 이름으로 언급됩니다.

관련 스레드. 지역 OS/390용 DB2 UDB 서브시스템에서 시작되어 원격 OS/390용 DB2 UDB 서브시스템에서 데이터에 액세스할 수 있는 스레드.

관련 주소 공간. OS/390용 DB2 UDB에서, OS/390용 DB2 UDB 외부에 있으면서 연결되어 있는 저장영역. 연합 주소 공간은 OS/390용 DB2 UDB 서비스를 요청할 수 있습니다.

관련된 바이트 주소(RBA). OS/390 환경에서, 데이터 세트나 속해 있는 파일에 할당되는 저장영역 공간의 맨 앞에서부터 데이터 레코드나 제어 가격에 대한 오프셋.

관리 지원 테이블. 이미지, 오디오 및 비디오 오브젝트에서 사용자 요청을 처리하기 위해 DB2 Extender에서 사용되는 테이블. 일부 관리 지원 테이블은 Extender에 대해 사용 가능한 사용자 테이블과 컬럼을 식별합니다. 다른 관리 지원 테이블에는 사용 가능하게 된 컬럼에 있는 오브젝트에 대한 속성 정보가 있습니다. 또한 *메타데이터 테이블*도 호출됩니다.

관리자 권한. 오브젝트 집합에 대한 사용자 특권을 제공하는 권한 레벨. 예를 들어, DBADM 권한은 데이터베이스내 모든 오브젝트에 대한 특권을 제공하고, SYSADM 권한은 시스템내 모든 오브젝트에 대한 특권을 제공합니다.

교착 상태. 트랜잭션이 다른 일부 트랜잭션에 의해 잠겨 있는 독점 자원에 종속되어 있고, 그 트랜잭션은 다시 원래 트랜잭션에 의해 사용되는 독점 자원에 종속되어 트랜잭션 처리가 불가능한 상태.

교착 상태 감지기. 교착 상태가 존재하는지 판별하기 위해 잠금 상태를 모니터하는 데이터베이스 관리 프로그램 내에서의 프로세스. 교착 상태가 감지되면 감지기는 교착 상태에 연루된 트랜잭션 중 하나를 중지시킵니다. 이 트랜잭션은 구간 복원되고 다른 트랜잭션이 진행됩니다.

구간 복원. SQL문에 의해 변경된 데이터를 마지막 확약점 상태로 복원하는 프로세스. *일관성 지점* 참조.

구간 복원중 실패. 복구 단위 상태. 복구 단위가 구간 복원되기 시작한 후 프로세스가 완료되기 전에 OS/390용 DB2 UDB가 실패하지만 할 경우, OS/390용 DB2 UDB는 재시작 동안 변경사항 백아웃을 계속합니다.

구문 문자 세트. IBM 레지스트리에 문자 세트 00640으로 등록된 81개의 그래픽 문자로 이루어진 세트. 이 세트는 원래 시스템과 국가 경계에서의 이식성과 교환 가능성을 최대화하기 위해 구문 목적으로 사용될 프로그래밍 언어 공유를 위한 것이었습니다. 이것은 몇 가지의 예외를 제외하고, 대부분의 1차 등록 문자 세트에 포함됩니다. 비교 - *불변 문자 세트*.

구별 유형. 내부적으로 기존의 유형(소스 유형)으로 표현되나 의미상 호환되지 않는 별도의 유형으로 간주되는 사용자 정의 데이터 유형.

구성원. OS/390 환경에서, 확약 프로세스에 참여하는 확약 조정자 이외의 다른 엔터티. 동의어 - SNA의 *에이전트*.

구성원. (1) DB2의 경우, *복사 작업 내역 세트 구성원*. (2) OLAP Starter 킷에서, 세 개 이상의 차원을 통해 데이터를 참조하는 메소드. 사실 테이블의 개인 데이터 값은 각 차원으로부터 한 구성원의 관계입니다.

구성원 범위. *명령 범위* 참조.

구성원 이름. 데이터 공유 그룹에서 특정 OS/390용 DB2 UDB 서브시스템에 대한 XCF 식별자.

구체화. OS/390용 DB2 UDB에서, (1) 뷰 또는 중첩 테이블 표현식에서 조회에 의한 추가 처리를 위한 작업 파일로 행을 기록하는 프로세스.

(2) LOB 값을 연속 저장영역에 놓는 것. LOB 값이 매우 클 수도 있으므로, OS/390용 DB2 UDB는 절대적으로 필요하게 될 때까지는 LOB 데이터를 실현하지 않습니다.

국가 코드. 데이터베이스에 액세스할 때 응용프로그램의 국가 코드를 사용하여 날짜 및 시간 프리젠테이션(표시 및 인쇄) 형식을 결정합니다. 또한, 코드 페이지와 함께 데이터베이스에 대한 기본 조합 순서를 결정하는 데에도 사용됩니다.

권한. *관리자 권한* 참조.

권한 부여. 권한 부여 ID에게 특권 또는 권한을 제공하는 것.

권한 부여 ID. (1) 권 집합을 지정하는 명령문 내의 문자열. 데이터베이스 관리 프로그램은, 테이블, 뷰, 색인 등과 같은 오브젝트 이름에 대한 내재적 규정자로서 권한 점검에 이를 사용합니다. (2) OS/390용 DB2 UDB에 대한 연결에 대해 확인할 수 있고 일련의 특권이 허용되는 문자열. 권한 부여 ID는 개별적 조직 그룹이나 함수를 나타낼 수 있지만 OS/390용 DB2 UDB는 이러한 표시를 판별하지 않습니다.

권한 부여된 프로그램 기능(APF). OS/390용 DB2 UDB에서, 제한된 함수들을 사용할 수 있는 권한이 부여된 프로그램의 식별을 허용하는 기능.

권한 취소. 권한 부여 ID로부터 권한 또는 특권을 제거하는 것.

규정. *CFRM 규정* 참조.

그래픽 문자. DBCS 문자.

그래픽 문자열. DBCS 문자 연속.

그룹. (1) 활동 또는 자원 액세스 권한에 따라 ID가 있는 사용자들의 논리적 조직. (2) Satellite Edition에서, 위성에서 수행하는 응용프로그램 및 데이터베이스 구성과 같이, 특성을 고유하는 위성 콜렉션.

그룹 버퍼 풀 양방향. OS/390 환경에서, 그룹 버퍼 풀 구조의 두 인스턴스에 데이터를 기록할 수 있는 능력(*1차 그룹 버퍼 풀* 및 *2차 그룹 버퍼 풀*). OS/390 서적에서는 이러한 인스턴스를 "이전"(1차의 경우) 및 "새"(2차의 경우) 구조라고 합니다.

그룹 범위. *명령 범위* 참조.

그룹 이름. OS/390 환경에서, 데이터 공유 그룹에 대한 XCF 식별자.

그룹 재시작. OS/390 환경에서, 잠금 또는 공유 통신 영역 유실 후에 데이터 공유 그룹의 최소한 한 구성원이 재시작하는 것.

글로벌 테이블 잠금. 테이블의 노드 그룹에 있는 모든 노드상에서 얻는 테이블 잠금.

글로벌 트랜잭션. 여러 개의 자원 관리 프로그램이 필요한 분산 트랜잭션 프로세싱 환경에서의 작업 단위(UOW).

기간 날짜. 연도, 달, 날짜를 나타내는 숫자의 DECIMAL(8,0) 값.

기간 시간. 시간, 분, 초를 나타내는 DECIMAL(6,0) 값.

기본 누적 테이블. DB2 복제에서, 소스 테이블이나 특정 시점 테이블로부터 일정한 간격으로 집계되는 데이터를 포함하는 목표 테이블의 유형.

기본 대화. APPC 기본 대화 API를 사용한 트랜잭션 프로그램 사이의 LU 6.2 변환. 반의어 - *맵 대화*.

기본 순차적 액세스 메소드(BSAM). 순차 액세스나 직접 액세스 장치를 사용하여 연속 순서대로 데이터 볼록을 저장하거나 검색하기 위해 OS/390용 DB2 UDB에서 사용하는 액세스 방법.

기본 술어. 두 값을 비교하는 술어.

기본 키. 테이블 정의의 일부인 고유 키. 기본 키는 참조 제한조건 정의의 기본 상위 키입니다.

기본 테이블. (1) CREATE TABLE문으로 작성되는 테이블. 그러한 테이블에는 데이터베이스에 물리적으로 저장된 데이터 및 설명 모두가 포함됩니다. 반의어 - *보기*. (2) OS/390용 DB2 UDB에서, (a) SQL CREATE TABLE문에 의해 작성되어 영구 데이터를 보유하는 테이블. 반의어 - *결과 테이블* 및 *임시 테이블*. (b) LOB 컬럼 정의를 포함하는 테이블. 실제 LOB 컬럼 데이터는 기본 테이블로 저장되지 않습니다. 기본 테이블에는 각 행에 대한 행 ID와 해당되는 각 LOB 컬럼에 대한 표시기 컬럼이 있습니다. 반의어 - *보조 테이블*.

기본 테이블 공간. OS/390용 DB2 UDB에서, 기본 테이블을 포함하는 테이블 공간.

기본 행수. 데이터베이스 테이블의 행 수.

기술 메타데이터. Data Warehouse Center에서, 데이터베이스 유형과 길이와 같은 데이터의 기술적 측면을 설명하는 데이터. 기술 메타데이터에는 데이터가 제공되는 장소와 데이터를 추출, 정리 및 변환하기 위해 사용되는 규칙에 대한 정보가 포함됩니다. Data Warehouse Center에 있는 많은 메타데이터가 기술 메타데이터입니다. 반의어 - *비지니스 메타데이터*.

기호 목적지 이름. 원격 상대 이름을 지정합니다. 이 이름은 클라이언트가 서버로의 APPC 연결을 설정하는 데 필요한 정보(상대 LU 이름, 모드 이름, 상대 TP 이름)가 들어 있는 CPI 통신 부속 정보 테이블에 있는 항목에 해당합니다.

길이 속성. 선언된 고정 길이 또는 최대 문자열 길이를 나타내는 문자열과 연관된 값.

끝 노드(EN). APPN에서, 지역 제어점과 인접한 네트워크 노드의 제어점 사이의 세션을 지원하는 노드.

나

날짜. 요일, 달, 연도를 나타내는 세 부분으로 구성된 값.

날짜시간 값. 데이터 유형 DATE, TIME 또는 TIMESTAMP의 값.

내보내기. 데이터베이스 관리 프로그램 테이블로부터 PC/IXF, DEL, WSF, ASC와 같은 형식을 사용하는 파일로 데이터를 복사하는 것. 반의어 - *가져오기*.

내부 자원 잠금 관리 프로그램(IRLM). OS/390 환경에서, OS/390용 DB2 UDB가 통신 및 데이터베이스 잠금을 제어하기 위해 사용하는 서브시스템.

내부 조인. 조인되는 모든 테이블들에 공통되지 않은 컬럼이 결과 테이블로부터 삭제되는 조인 방법. 반의어 - *외부 조인*.

내부 CCD 테이블. 직접 복사 작업을 할 수 없는 CCD 테이블. 이 테이블은 등록 테이블에 해당되는 고유 행이 없으므로, 연관된 복제 소스에 대한 행에서 CCD_OWNER 및 CCD_TABLE로 참조됩니다. 반의어 - **외부 CCD 테이블**.

내장 함수. DB2에 의해 제공되며 SYSIBM 스키마에 나타나는 SQL 함수. 반의어 - *사용자 정의 함수*.

내재된 특권. 동의어를 삭제할 수 있는 특권과 같이 오브젝트 소유권을 동반하거나, 유틸리티 작업을 종료할 수 있는 SYSADM 권한의 특권과 같이 권한을 보유하는 특권.

널(NULL). OS/390용 DB2 UDB에서, 정보가 없음을 나타내는 값.

널(NULL) 값. 값이 지정되지 않는 매개변수 위치.

널(NULL) 입력 가능. 컬럼의 값, 함수 매개변수 또는 결과가 값 부재를 수반할 수 있는 상태. 예를 들어, 개인 중간 이름의 첫 문자 필드에 반드시 값을 입력할 필요가 없으면 그 필드는 널(NULL) 입력 가능으로 간주됩니다.

네트워크 규정 이름. 상호 연결된 SNA 네트워크를 통해 LU가 알려져 있는 이름. 네트워크 규정 이름은 네트워크 LU 이름과, 개별 부속 네트워크를 식별하는 네트워크 이름으로 구성됩니다. 네트워크 규정 이름은 상호연결된 네트워크를 통해 고유합니다. *네트워크 규정 LU 이름*이나 *완전한 LU 이름*으로도 알려져 있습니다.

네트워크 노드 서버. 지역 논리 장치 및 인접 끝 노드에 네트워크 서비스를 제공하는 APPN 네트워크 노드.

네트워크 노드(NN). APPN에서, 다른 APPN 네트워크 노드, 세션 및 경로지정 서비스와의 분산 디렉토리 서비스, 토폴로지 데이터베이스 교환을 제공하는 네트워크상의 노드. 동의어 - *APPN 네트워크 노드*.

네트워크 디렉토리 서비스(NDS). 네트워크상의 모든 자원에 대한 정보를 유지보수하고 그 정보에 대한 액세스를 제공하는 분산된 전역 복제 데이터베이스 NerWare. NetWare 디렉토리 데이터베이스는 물리적 위치와 상관 없이, 디렉토리 트리라는 계층적 트리 구조로 오브젝트를 구성합니다.

네트워크 서비스. SSCP간, SSCP와 PU간, SSCP와 LU간, CP간 세션을 통해 네트워크 조작을 제어하는 네트워크 주소지정 가능 단위 내의 서비스.

네트워크 식별자(NID). OS/390 환경에서, IMS 또는 CICS에 의해 지정되는 네트워크 ID나, 연결 유형이 RRSAF일 경우, OS/390 RRS 복구 단위 ID(URID).

네트워크 이름. SNA에서, 일반 사용자가 네트워크 주소지정 장치(NAU), 링크 스테이션 또는 링크를 참조하는 기호 이름. 동의어 - *NETID*.

네트워크 주소. 네트워크에서의 노드 식별자.

네트워크 주소 지정가능 장치(NAU). 경로 제어 네트워크에 의해 전송되는 정보의 원점 또는 목적지. NAU는 논리 장치(LU), 물리 장치(PU), 제어점(CP) 또는 시스템 서비스 제어점(SSCP)이 될 수 있습니다. 또한 *네트워크 이름* 참조.

노드. (1) 데이터베이스 파티션에서, 데이터베이스 파티션의 동의어. (2) 하드웨어에서, 대량 병렬 처리(MPP) 시스템 또는 클러스터 시스템의 일부인 단일 프로세서 또는 대칭 멀티 프로세서(SMP) 컴퓨터. 예를 들어, RS/6000® SP™는 고속 네트워크에 의해 연결되는 여러 노드들로 구성되는 MPP 시스템입니다. (3) 통신에 있어, 네트워크 내에서 두 개 이상의 링크에 공통된 통신 링크 또는 연결의 끝점. 프로세서, 통신 제어기, 클러스터 제어기, 터미널, 워크스테이션 등이 노드가 될 수 있습니다. 노드는 경로지정이나 다른 기능에 따라 다양합니다.

노드 그룹. 하나 이상의 데이터베이스 파티션으로 구성된 명명된 그룹.

노드 디렉토리. 클라이언트 워크스테이션에서 모든 적용가능 데이터베이스 서버로의 통신을 설정하는 데 필요한 정보가 들어 있는 디렉토리.

논리 노드. 여러 개의 노드가 지정된 프로세서상의 노드. 또한 *논리* 참조.

논리 복구 보류(LRECP). OS/390용 DB2 UDB에서, 데이터를 참조하는 색인 키나 데이터가 불일치되는 상태.

논리 색인 파티션. OS/390용 DB2 UDB에서, 같은 데이터 파티션을 참조하는 모든 키의 세트.

논리 연산자. 검색 조건(AND, OR)을 얼마나 많이 평가할 것인지 또는 검색 조건의 논리 센서를 변환할 것인지(NOT) 지정하는 키워드.

논리 작업 단위 식별자(LUWID). OS/390 환경에서, 네트워크 내에서 스레드를 고유하게 식별하는 이름. 이 이름은 전체적으로 규정된 LU 네트워크 이름, LUW 인스턴스 번호 및 LUW 순차 번호로 구성됩니다.

논리 작업 단위(LUW). 프로그램이 동기화점 사이에 수행하는 처리.

논리 잠금(L-잠금). OS/390용 DB2 UDB에서, 트랜잭션이 트랜잭션 사이의 DB2간 데이터 동시성을 제어하기 위해 사용하는 잠금 유형. 반의어 - *물리적 잠금*.

논리 장치 6.2(LU 6.2). APPC를 사용하여 두 응용프로그램간의 세션을 지원하는 LU 유형.

논리 장치(LU). (1) SNA에서, 다른 일반 사용자와 통신하기 위해 SNA 네트워크에 액세스하게 되는 포트. LU는 다른 LU가 있는 많은 세션을 지원할 수 있습니다. (2) OS/390 환경에서, 응용프로그램이 다른 응용프로그램과 통신하기 위해 SNA 네트워크에 액세스하는 액세스점. 또한 *LU 이름* 참조.

논리 파티션. OS/390용 DB2 UDB에서, 특정 파티션과 연관되는 비파티션 색인의 키 또는 RID 쌍 세트.

논리 페이지 목록(LPL). OS/390용 DB2 UDB에서, 오류가 있어서 페이지가 분석될 때까지 응용프로그램에 의해 참조될 수 없는 페이지들의 목록. 실제 매체(결합 기능 또는 DASD)에 오류가 포함되어 있지 않을 수도 있으므로, 페이지 오류는 논리 오류입니다. 보통 매체에 대한 연결이 유실됩니다.

논리적 드레인. OS/390용 DB2 UDB에서, 비파티션 색인의 논리적 파티션에서의 드레인.

논리적 청구. OS/390용 DB2 UDB에서, 비파티션 색인의 논리적 파티션에서의 청구.

다

다중 사이트 갱신. OS/390용 DB2 UDB에서, 단일 작업 단위(UOW) 내의 여러 위치에서 데이터가 갱신되는 분산 관계형 데이터베이스 처리.

다중 차원. OLAP Starter 킷에서, 세 개 이상의 차원을 통해 데이터를 참조하는 메소드. 사실 테이블의 개인 데이터 값은 각 차원으로부터 한 구성원의 관계입니다.

다중 차원 데이터베이스. OLAP Starter 킷에서, OLAP 분석을 위해 관계형 데이터를 복사하는 비관계형 데이터베이스.

단계. Data Warehouse Center에서, 웨어하우스 프로세스의 단일 조작 데이터. 대부분의 경우, 단계에는 웨어하우스 소스, 데이터 전송 또는 이동에 대한 설명, 그리고 목표가 포함됩니다. 단계는 스케줄에 따라 실행되거나, 다음 단계에서 연속될 수 있습니다.

단순 테이블 공간. OS/390용 DB2 UDB에서, 파티션되지도 않고 세그먼트화되지도 않은 테이블 공간.

단순 페이지 세트. OS/390용 DB2 UDB에서, 파티션되지 않은 페이지 세트. 단순 페이지 세트는 처음에 단일 페이지 세트(페이지 세트 조각)로 구성됩니다. 데이터 세트는 2 GB로 확장되면, 또 다른 데이터 세트가 작성되고, 이렇게 총 32개의 데이터 세트까지 작성됩니다. OS/390용 DB2 UDB는 데이터 세트를, 최대 64 GB를 포함하는 단일 연속 선형 주소 공간인 것으로 간주합니다. 데이터는 파티션 스킴에 관계 없이 이 주소 공간 내에서 다음으로 사용 가능한 위치에 저장됩니다.

단정밀도 부동 소수점 수. 실수에 대한 대략적인 32 비트 표시.

대기행렬에 대기된 순차적 액세스 메소드(QSAM). 기본 순차적 액세스 메소드(BSAM)의 확장 버전. 이 방법이 사용될 경우, 대기행렬은 보조 저장영역이나 출력 장치로 전송되기를 기다리는 출력 데이터나 처리를 기다리는 입력 데이터 블록으로 형성됩니다.

대체 문자. SQL에서, 목표 코딩 표시에서 일치되는 것이 없는 소스 프로그램의 문자에 대한 문자 변환 동안 대체되는 고유한 문자.

대형 오브젝트(LOB). 길이가 최고 2 GB인 일련의 바이트들. BLOB(2진), CLOB(1바이트 문자 또는 혼합 문자), DBCLOB(2바이트 문자), 세 가지 유형이 있습니다.

대형 잠금. OS/390용 DB2 UDB에서, 테이블, 파티션 또는 테이블 공간에 대한 *공유*, *갱신* 또는 *실행* 모드 잠금.

대형 테이블 공간. 대형 오브젝트(LOB) 데이터 또는 LONG 문자열만 저장할 수 있는 테이블 공간.

대화. APPC에서, 트랜잭션을 처리하는 동안 서로 통신할 수 있도록 하는 논리적 장치간(LU-to-LU) 세션을 통하는 두 트랜잭션 프로그램 사이의 연결.

대화 보안. APPC에서, 연결을 설정하기 전에 사용자 ID 또는 그룹 ID 및 암호 확인을 허용하는 프로세스.

대화 보안 프로파일. 대화 보안을 위해 APPC가 사용하는 사용자 ID 또는 그룹 ID 및 암호 집합.

대화식 트랜잭션. APPC에서, 논리 장치(LU) 서비스를 사용한 두 개 이상의 프로그램 통신.

데이터 공간. OS/390용 DB2 UDB에서, 프로그램이 직접 조작할 수 있는 2 기가바이트까지의 연속 가상 저장영역 주소 범위. 주소 공간과는 달리, 데이터 공간은 데이터만 보유할 수 있고, 공통 영역, 시스템 데이터 또는 프로그램은 포함하지 않습니다.

데이터 공유. 단일 데이터 세트에 직접 액세스하여 변경할 수 있는 두 개 이상의 OS/390용 DB2 UDB 서브시스템 능력.

데이터 공유 구성원. XCF 서비스에 의해 데이터 공유 그룹에 지정된 OS/390용 DB2 UDB 서브시스템.

데이터 공유 그룹. 데이터 무결성을 유지보수하는 동안 같은 데이터에 직접 액세스하여 변경하는 하나 이상의 OS/390용 DB2 UDB 서브시스템 콜렉션.

데이터 링크 제어(DLC). SNA에서, 두 노드 사이의 링크를 통해 데이터 전송을 스케줄하고 링크에 대한 오류 제어를 수행하는 링크 스테이션으로 구성된 프로토콜 계층.

데이터 마트. 부서나 팀의 특정 요구사항에 맞게 조정된 데이터가 있는 데이터 웨어하우스 부속 집합. 데이터 마트는 OLAP 도구에 포함된 데이터와 같이, 전체 조직을 위한 웨어하우스 부속 집합이 될 수 있습니다.

데이터 블로킹. 복사 작업 내역 순환 동안 변경 데이터의 어느 정도가 복제되는 지를 지정하는 프로세스. 반의어 - *블로킹*.

데이터 설명 언어. 동의어 - *데이터 정의 언어*.

데이터 영역. 프로그램이 정보를 보유하는 데 사용하는 메모리 영역.

데이터 유형. SQL에서, 컬럼, 리터럴, 호스트 변수, 특수 레지스터, 그리고 함수 및 표현식의 결과에 대한 속성.

데이터 일관 변경(CCD) 테이블. DB2 복제에서, 데이터를 감사하거나 스테이징하기 위해, 또는 둘 다를 수행하기 위해 사용되는 목표 테이블 유형. **완료 CCD 테이블**, **압축 CCD 테이블**, *외부 CCD 테이블*, *내부 CCD 테이블*, **비완료 CCD 테이블**, **비압축 CCD 테이블**도 참조.

데이터 정의 이름(ddname). OS/390용 DB2 UDB에서, 같은 이름을 포함하는 데이터 제어 블록에 해당되는 데이터 정의(DD) 명령문의 이름.

데이터 파티션. OS/390 환경에서, 파티션된 테이블 공간내에 있는 VSAM 데이터 세트.

데이터 현재성. OS/390용 DB2 UDB에서, 프로그램의 호스트 변수 내로 검색되는 데이터가 기본 테이블의 데이터 사본인 상태.

데이터베이스 관리 공간(DMS) 테이블 공간. 그 공간이 데이터베이스에 의해 관리되는 테이블 공간. 반의어 - *SMS 테이블 공간*.

데이터베이스 관리 시스템(DBMS). 동의어 - *데이터베이스 관리 프로그램*.

데이터베이스 관리 프로그램. 효율적인 액세스, 무결성, 복구, 동시처리 제어, 독립성 및 보안을 위한 복합 물리적 구조, 중앙 제어 및 데이터 독립성 서비스를 제공함으로써 데이터를 관리하는 컴퓨터 프로그램.

데이터베이스 관리 프로그램 인스턴스. 물리적 데이터베이스 관리 프로그램 환경 이미지와 유사한 논리 데이터베이스 관리 프로그램 환경. 동일한 워크스테이션에서 여러 개의 데이터베이스 관리 프로그램 제품 인스턴스가 있을 수 있습니다. 이들 인스턴스를 상용하여 제품 환경으로부터 개발 환경을 분리하고, 데이터베이스 관리 프로그램을 특정 환경으로 조정하며, 특정 그룹의 사람들로부터 민감한 정보를 보호할 수 있습니다.

데이터베이스 관리자(DBA). 데이터베이스의 설계, 개발, 조작, 보호, 유지보수 및 사용에 대한 책임을 맡고 있는 사람.

데이터베이스 노드. *데이터베이스 파티션* 참조.

데이터베이스 디렉토리. 클라이언트가 연결할 수 있는 모든 데이터베이스에 대한 데이터베이스 액세스 정보가 들어 있는 디렉토리.

데이터베이스 로그. 데이터베이스에 대한 모든 변경사항을 기록하는 로그 레코드로 구성된 1차 및 2차 로그 파일 집합. 데이터베이스 로그를 사용하여 확약되지 않은 작업 단위에 대한 변경사항을 구간 복원하고 일관된 상태로 데이터베이스를 복구할 수 있습니다.

데이터베이스 서버. 데이터베이스에 대한 데이터베이스 서비스를 제공하는 기능 장치.

데이터베이스 설명자(DBD). OS/390용 DB2 UDB 데이터베이스 정의의 내부적 표시로, OS/390용 DB2 UDB 카탈로그에 있는 데이터 정의를 반영합니다. 데이터베이스 스크립터에 정의된 오브젝트는 테이블 공간, 테이블, 색인, 색인 공간 및 관계입니다.

데이터베이스 시스템 모니터. 데이터베이스 관리 프로그램 및 DB2 Connect을 사용하여 데이터베이스 관리 프로그램, 데이터베이스, 응용프로그램에 대한 상태 정보 및 성능을 모니터하는 프로그래밍 API 콜렉션.

데이터베이스 액세스 스레드. OS/390용 DB2 UDB에서, 원격 서브시스템 대신 지역 서브시스템에서 데이터에 액세스하는 스레드.

데이터베이스 엔진. 데이터베이스 사용에 필요한 기본 함수 및 구성 파일을 제공하는 데이터베이스 관리 프로그램의 일부.

데이터베이스 연결 서비스(DCS) 디렉토리. 응용프로그램 리퀘스터에 액세스하는 데 사용되는 원격 데이터베이스 및 해당 응용프로그램 리퀘스터에 대한 항목이 들어 있는 디렉토리.

데이터베이스 오브젝트. SQL로 작성 또는 조작될 수 있는 모든 것(예. 테이블, 뷰, 색인, 패키지, 트리거 또는 테이블 공간).

데이터베이스 요청 모듈(DBRM). OS/390용 DB2 UDB 사전 처리 컴파일러에 의해 작성되고 SQL문에 대한 정보를 포함하는 데이터 세트 구성원. DBRMs는 바인드 프로세스에서 사용됩니다.

데이터베이스 응용프로그램 원격 인터페이스(DARI). *저장 프로시듀어*에 대한 사용되지 않는 용어.

데이터베이스 카탈로그. Data Warehouse Center에서, 테이블, 뷰 및 색인과 같은 데이터베이스 오브젝트 설명을 포함하는 테이블 콜렉션.

데이터베이스 클라이언트. 데이터베이스 서버에 있는 데이터베이스에 액세스하기 위해 사용되는 워크스테이션.

데이터베이스 파티션. 사용자 자신의 데이터, 색인, 구성 파일, 트랜잭션 로그로 구성된 데이터베이스의 일부. 간혹 노드 또는 *데이터베이스 노드*라고 합니다.

도메인 이름. TCP/IP 응용프로그램이 TCP/IP 네트워크 내에서 TCP/IP 호스트를 참조하는 이름. 도메인 이름은 도트로 구분되는 연속된 이름들로 구성됩니다.

도메인 이름 서버(DNS). TCP/IP 호스트 이름을 IP 주소에 맵핑하기 위해 사용되는 분산 디렉토리를 관리하는 TCP/IP 네트워크 서버.

도메인 이름 시스템. 인간이 읽을 수 있는 머신 이름을 IP 주소로 맵핑하는 TCP/IP에 의해 사용되는 분산 데이터베이스 시스템.

독립. OS/390용 DB2 UDB에서, 다른 오브젝트의 상위도 아니고, 종속되어 있지도 않은 오브젝트(행, 테이블 또는 테이블 공간).

독립 논리 장치(ILU). 시스템 서비스 제어점(SSCP)으로부터의 보조 없이 LU간 세션을 활성화할 수 있는 논리 장치. ILU에는 SSCP-to-LU 세션이 없습니다. 반의어 - *종속 논리 단위*.

독립형. OS/390용 DB2 UDB 서비스를 사용하지 않고, OS/390용 DB2 UDB와 별도로 실행될 수 있음을 의미하는 프로그램 속성.

독점 잠금. 현재 실행중인 응용프로그램 프로세스가 데이터베이스 데이터를 액세스하지 못하도록 하는 잠금.

동기. 공통 타이밍 신호와 같은 특정 이벤트 발생에 의존하는 두 개 이상의 프로세스와 관련된 용어. 반의어 - *비동기*.

동기점. *일관성 지점* 참조.

동기화 수준. APPC에서, 해당 트랜잭션 프로그램이 확정 요청 및 응답을 교환하는지 표시하는 스펙.

동시성. 동시에 여러 명의 대화시 사용자나 적용업무 프로세스가 자원을 사용하는 것.

동의어. OS/390용 DB2 UDB에서, 테이블이나 뷰에 대한 SQL로 된 대체 이름. 동의어는 동의어가 정의된 서브시스템에 있는 오브젝트를 참조하기 위해서만 사용될 수 있습니다.

동적 바인드. SQL문이 입력될 때 바인드되는 프로세스. 또한 *바인드* 참조.

동적 SQL. 수행중인 프로그램 내에서 준비 및 실행되는 SQL문. 동적 SQL에서, SQL 소스는 프로그램으로 코드화되지 않고 호스트 언어 변수에 들어 있습니다. 프로그램이 수행되는 동안 SQL문은 여러 번 변경될 수 있습니다.

드레인. OS/390용 DB2 UDB에서, 해당되는 오브젝트에 대한 액세스를 Quiesce하여 잠긴 자원을 확보하는 것.

드레인 잠금. OS/390용 DB2 UDB에서, 청구가 발생하지 않도록 하는 청구 클래스에 대한 잠금.

등록. *복제 소스* 참조.

등록 정보. Data Warehouse Center에서, 정보 단위를 설명하는 특성이나 속성. 각 오브젝트 유형에는 연관되는 등록 정보 세트가 있습니다. 각 오브젝트마다 등록 정보에 값 세트가 지정됩니다.

등록 프로세스. DB2 복제에서, 복제 소스를 정의하는 프로세스. 반의어 - **복사 작업 정의 프로세스**.

디렉토리. 데이터베이스 설명 및 윤곽 커서 테이블과 같은 내부 오브젝트를 포함하는 OS/390용 DB2 UDB 시스템 데이터베이스.

디렉토리 서비스. APPN 네트워크의 자원 위치에 대한 정보를 유지보수하는 APPN 프로토콜 부분.

라

래치. 동시 이벤트나 시스템 자원의 사용을 제어하기 위한 OS/390용 DB2 UDB 내부 메카니즘.

랩퍼. 연합 데이터베이스 시스템에서, 연합 서버가 데이터 소스와 통신하여 데이터를 검색하기 위해 루틴을 호출하는 메카니즘. 루틴은 **랩퍼 모듈**이라고 하는 라이브러리에 있습니다.

레이블된 지속 기간. 연도, 개월, 날짜, 시간, 분, 초 또는 마이크로세컨드를 나타내는 숫자.

레지스트리 데이터베이스. OS/390 환경에서, 핵심부, 그룹, 조직, 계정 및 보안 규정에 대한 보안 정보에 대한 데이터베이스. DCE 보안 구성요소는 레지스트리 데이터베이스를 유지보수합니다.

레코드. 테이블이나 다른 데이터의 단일 행에 대한 저장영역 표시.

레코드 식별자(RID). 테이블에 있는 레코드를 고유하게 식별하기 위해 DB2가 내부적으로 사용하는 번호. RID에는 레코드가 저장되는 페이지를 주소지정하기 위한 충분한 정보가 있습니다. 비교 - *행 ID*.

레코드 식별자(RID) 풀. OS/390용 DB2 UDB에서, 목록 프리페치 처리 동안 레코드 ID를 정렬하기 위해 보존되는 16 MB 라인 이상의 주 저장영역.

레코딩. 차후에 볼 수 있는 성능 스냅샷의 정보.

로그. (1) 시스템에서의 변경사항을 기록하는 데 사용되는 파일. (2) OS/390용 DB2 UDB 실행 동안 발생하는 이벤트를 설명하고 그 이벤트들의 순서를 나타내는 레코드 콜렉션. 그러므로, 기록되는 정보는 OS/390용 DB2 UDB 실행 동안 실패하는 이벤트가 발생할 때 복구에 사용됩니다. (3) *데이터베이스 로그* 참조.

로그 레코드. 작업 단위중에 수행되는 데이터베이스에 대한 갱신 레코드. 이 레코드는 사용중인 로그의 최신 로그 레코드 이후에 작성됩니다.

로그 레코드 순차 번호(LRSN). OS/390용 DB2 UDB가 생성하여 각각의 로그 레코드와 연관시키는 번호. LRSN은 또한 페이지 버전화 작업에도 사용됩니다. 특정의 OS/390용 DB2 UDB 데이터 공유 그룹에서 생성하는 LRSN은 각 DB2 로그마다 순서가 확실하게 증가하고 데이터 공유 그룹의 각 페이지마다 순서가 확실하게 증가하게 합니다.

로그 절단. OS/390용 DB2 UDB에서, 명시적으로 시작하는 RBA가 설정되는 프로세스. 이 RBA는 로그 데이터의 다음 바이트가 기록될 지점입니다.

로그 초기설정. OS/390용 DB2 UDB가 로그의 현재 끝을 찾으려고 시도하는 재시작 처리의 첫번째 단계.

로그 테이블. 색인화되는 텍스트 문서에 대한 정보를 포함하는, Text Extender에 의해 작성된 테이블.

로그 파티션. 해당 데이터베이스 파티션에 대한 데이터베이스 활동을 기록하는 각 데이터베이스 파티션상의 로그 파일.

로그 헤드. 사용중인 로그에서 가장 오래전에 작성된 로그 레코드.

로드 모듈. 실행을 위해 주 저장영역에 로드하기에 적합한 프로그램 단위. 링크 편집기의 출력.

로드 복사. 데이터의 백업 이미지는 이전에 로드되었고 롤 포워드 복구중에 복원될 수 있습니다.

로드 유틸리티. 테이블 데이터의 블록 갱신을 수행하는 비트랜잭션 유틸리티. 반의어 - *가져오기 유틸리티*.

로케일. OS/390용 DB2 UDB에서, 특정 언어와 국가, 그리고 CCSID에 대해 정의된 문자를 결합하는 사용자 환경 부속 집합의 정의.

롤 포워드. 데이터베이스 로그에 기록된 변경사항을 적용함으로써, 복원된 데이터베이스에 있는 데이터를 갱신하는 프로세스. *포워드 복구* 참조.

루트 페이지. OS/390용 DB2 UDB에서, 첫번째 색인 공간 맵 페이지 다음에 오는 색인 페이지 세트의 페이지. 루트 페이지는 색인의 최상위 레벨(또는 시작점)입니다.

루틴. OS/390용 DB2 UDB에서, 사용자 정의 함수 또는 저장 프로시듀어.

리바인드. 이전에 바인드된 응용프로그램의 패키지를 작성하는 것. 예를 들어, 프로그램에 의해 액세스되는 테이블에 대해 하나의 색인이 추가될 경우, 새로운 색인을 이용할 수 있도록 그 테이블에 패키지가 다시 바인드되어야 합니다.

리퀘스터. OS/390용 DB2 UDB에서, 원격 RDBMS에 대한 요청의 소스로, 데이터를 요청하는 시스템. 동의어 - *응용프로그램 리퀘스터.*

리프 페이지. OS/390용 DB2 UDB에서, 키 및 RID의 쌍을 포함하고 실제 데이터를 지시하는 페이지. 반의어 - *비 리프 페이지.*

링크 편집. OS/390용 DB2 UDB에서, 링크 편집기를 사용하여 로드 가능한 컴퓨터 프로그램을 작성하는 조치.

링크 편집기. 모듈 사이의 상호 참조를 분석하고 필요에 따라 주소를 조정하여 하나 이상의 오브젝트 모듈이나 로드 모듈에서 로드 모듈을 작성하기 위한 컴퓨터 프로그램.

마

마스킹 문자. 검색 용어의 앞부분, 중간 부분, 끝 부분에서 선택적 문자를 나타내는 데 사용하는 문자. 정확한 색인에서 용어의 변이형을 찾기 위해 일반적으로 마스킹 문자들이 사용됩니다.

매개변수 표시문자. 동적 SQL문의 명령문 문자열에 나타나는 물음표(?). 물음표는 명령문 문자열이 정적 SQL문일 경우 호스트 변수가 나타날 수 있는 곳에 표시될 수 있습니다.

매개변수화된 데이터 유형. 특정 길이, 스케일 또는 정밀도로 정의될 수 있는 데이터 유형. 문자열 및 십진수 데이터 유형들에 대해 매개변수가 작성됩니다.

맵 대화. APPC에서, APPC 맵 대화 API를 사용한 두 트랜잭션 프로그램(TP) 사이의 대화. 일반적인 상황에서, 일반 사용자 TP는 맵 대화를 사용하고 서비스 TP는 기본 대화를 사용합니다. 두 유형 중 한 유형의 프로그램이 두 유형의 대화 중 하나를 사용할 수 있습니다. 반의어 - *기본 대화.*

멀티타스킹. 두 개 이상의 타스크에 대해 동시 처리 성능이나 인터리브 실행을 제공하는 조작 모드.

메뉴. OS/390용 DB2 UDB에서, 운영자가 선택할 수 있는 함수들을 표시하는 목록. 메뉴는 간혹 *메뉴 패널*이라고 합니다.

메타데이터. 저장 데이터, 설명적 데이터의 특성을 설명하는 데이터. 예를 들어, 데이터베이스 테이블에 대한 메타데이터에는 테이블의 이름, 테이블을 포함하는 데이터베이스의 이름, 테이블에 있는 컬럼의 이름, 그리고 기술 용어나 비지니스 용어를 사용한 컬럼 설명이 포함될 수 있습니다.

메타데이터 가져오기. 메타데이터를 동적(사용자 인터페이스를 통해)이나 일괄처리로 Data Warehouse Center에 가져오는 프로세스.

메타데이터 문서 프로세스. 원래의 메타데이터와 동기화되어 발행된 메타데이터를 보존하기 위해 발행 후에 작성된 모든 단계를 포함하는, Data Warehouse Center에 의해 작성된 프로세스.

명령. OS/390용 DB2 UDB 연산자 명령 또는 DSN 부속명령. 명령은 SQL문에서 구별됩니다.

명령 범위. OS/390용 DB2 UDB에서, 데이터 공유 그룹에서 명령 조작 범위. 명령에 *구성원 범위*가 있을 경우, 그 명령은 한 구성원의 정보만 표시하거나 그 구성원이 지역적으로 소유하는 비공유 자원들에만 영향을 줍니다. 명령에 *그룹 범위*가 있을 경우, 그 명령은 모든 구성원의 정보만 표시하거나 모든 구성원이 지역적으로 소유하는 비공유 자원들에 영향을 주거나, 공유 가능한 자원에 영향을 줍니다.

명령 인식 문자(CRC). MVS 콘솔 운영자나 IMS 서브시스템 사용자가 DB2 명령을 특정의 OS/390용 DB2 UDB 서브시스템으로 보낼 수 있게 하는 문자.

명령 접두부. OS/390용 DB2 UDB에서, 1 - 8자의 명령 식별자. 명령 접두부는 명령을 OS/390보다는 응용프로그램이나 서브시스템에 속하는 것으로 구별합니다.

명령문. 프로그램이나 프로시듀어 내에서의 지시.

명령문 문자열. OS/390용 DB2 UDB 환경에서의 동적 SQL문의 경우, 명령문의 문자열 양식.

명령문 트리거. OS/390용 DB2 UDB에서, 트리거 수준 FOR EACH STATEMENT로 정의된 트리거.

명령문 핸들. CLI에서, SQL문에 대한 정보가 들어 있는 데이터 오브젝트를 참조하는 핸들. 여기에는 동적 인수, 동적 인수 및 컬럼 바인딩, 커서 정보, 결과 값, 상태 정보와 같은 정보가 포함됩니다. 각 명령문 핸들은 연결 핸들과 연관되어 있습니다.

명령행 처리기(CLP). SQL문 및 데이터베이스 관리 프로그램 명령을 입력하기 위한 문자 기초 인터페이스.

명백한 유형 지정. OS/390용 DB2 UDB에서, 구별 유형에 정의된 사용자 정의 함수와 조작만 그 유형에 적용될 수 있도록 보증하는 프로세스. 예를 들어, 캐나다 달러와 미국 달러와 같이, 두 유형의 통화를 직접 비교할 수 없습니다. 그러나, 사용자 정의 함수(UDF)를 제공하여 하나의 통화를 다른 통화로 변환한 후 비교를 수행할 수 있습니다.

명시적 계층 구조 잠금. OS/390용 DB2 UDB에서, IRLM에 알려진 자원들 사이에 상위-하위 관계를 만들기 위해 사용되는 잠금. 이러한 종류의 잠금을 사용하면 자원에 대해 어떤 DB2간 관계도 없을 때 전역 잠금 오버헤드를 피할 수 있습니다.

명시적 특권. SELECT 특권처럼, 이름이 있고 SQL GRANT 및 REVOKE 명령문 결과로 보유되는 특권. 반의어 - *내재된 특권*.

명확한 커서. 관계형 데이터베이스가 블로킹이 응답자 세트에서 사용될 수 있는 지를 판별할 수 있게 하는 커서. FOR FETCH ONLY 또는 FOR READ ONLY로 정의된 커서는 블로킹과 함께 사용될 수 있는 반면, FOR UPDATE로 정의된 커서는 그렇지 않습니다.

모니터 세션. 이전에 모니터된 데이터베이스 관리 프로그램으로부터 정보를 재생하거나 데이터베이스 관리 프로그램을 모니터하는 것. DB2 성능 모니터을 사용하여 어떤 데이터베이스 오브젝트를 모니터할 것인지 선택하고 모니터 세션을 작성합니다.

모니터 스위치. 성능 스냅샷에서 리턴되는 정보량 및 정보 유형을 제어하기 위해 사용자가 조작하는 데이터베이스 관리 프로그램 매개변수.

모델 통계. SQL문에서 참조되거나 참조되지 않을 수 있는 데이터베이스 오브젝트 통계로서, Explain 모델에 현재 존재합니다. 오브젝트는 현재 데이터베이스에 있을 수도 없을 수도 있습니다.

모드. Data Warehouse Center에서, 개발, 테스트 또는 생산과 같은 개발 단계.

모드 이름. (1) APPC에서, 메시지 길이 한계, 동기점, 전송 네트워크 내부의 서비스 클래스, 세션 경로지정 및 지연 특성과 같이, 세션에 요구되는 특성을 지정하기 위해 세션 초기설정자가 사용하는 이름. (2) OS/390 환경에서, 세션의 물리적 및 논리적 특성과 속성의 콜렉션에 대한 VTAM 이름.

목록 구조. OS/390 환경에서, 데이터를 대기행렬의 요소로 공유하거나 조작할 수 있게 하는 결합 기능 구조.

목록 프리페치. 순차적으로 데이터에 액세스하지 않는 조회에서도 프리페치를 이용하는 액세스 방법. 이것은 데이터 페이지에 액세스하기 전에 RID를 수집하고 색인을 스캔함으로써 이루어집니다. 그런 후 이들 RID가 정렬되고 이 목록을 사용하여 데이터가 사전에 페치됩니다.

목표. Data Warehouse Center에서, 한 단계에 의해 생성되거나 처리되는 테이블, 뷰 또는 파일. 단계의 출력.

목표 서버. DB2 복제에서, 목표 테이블의 데이터베이스 위치. 일반적으로 이 서버는 Apply 프로그램의 위치이기도 합니다.

목표 테이블. DB2 복제에서, 데이터가 복사되는 목표 서버의 테이블. 이는 사용자 복사 테이블, 특정 시점 테이블, 기본 총계 테이블, 변경 총계 테이블, 데이터 일관 변경 테이블 또는 Replica 테이블이 될 수 있습니다.

문서 모델. 포함하는 절의 측면에서 본 문서 구조의 정의. Text Extender는 색인화할 때 문서 모델을 사용합니다.

문자 대형 오브젝트(CLOB). 2 기가바이트까지의 일련의 문자(1 바이트, 멀티바이트 또는 둘 다). CLOB는 대형 텍스트 오브젝트를 저장할 수 있습니다. 이를 또한 문자 대형 오브젝트 문자열이라고도 합니다. 비교 - *2진 대형 오브젝트 (BLOB).*

문자 데이터 표현 아키텍처(CDRA). 일관된 표시 보존, 처리 및 문자열 데이터의 상호 교환에 사용되는 아키텍처.

문자열. 프로그래밍 언어에서, 텍스트 저장 및 조작에 사용되는 데이터 형식.

문자열. 연속된 바이트 또는 문자.

문자열 분리 문자. 가져오거나 내보내는 컬럼 식별자가 있는 ASCII 파일에 문자열을 넣는 데 사용되는 문자. *분리문자* 참조.

물리 장치(PU). SSCP과 PU간 세션을 통한 SSCP에서의 요청에 따라, 노드와 연관되는 자원(접속된 링크 및 인접한 링크 스테이션과 같은)을 관리하고 모니터하는 구성요소. SSCP는 접속된 링크와 같은 노드의 자원을 PU를 통해 간접적으로 관리하기 위해 PU에서 세션을 활성화합니다. 이 용어는 유형 2.0, 4, 5 노드에만 적용됩니다. 또한 *제어점* 참조.

물리적 드레인. OS/390용 DB2 UDB에서, 전체 비파티션 색인에 대한 드레인.

물리적 일관성. OS/390용 DB2 UDB에서, 부분적으로 변경된 상태에 있지 않은 페이지.

물리적 잠금 경합. OS/390용 DB2 UDB에서, 물리적 잠금에 대한 리퀘스터의 충돌 상태. *조정 가능 잠금*도 참조.

물리적 잠금(P-잠금). OS/390용 DB2 UDB가 다른 OS/390용 DB2 UDB 서브시스템에서 캐쉬되는 데이터의 일관성을 제공하기 위해 확보하는 잠금. 물리적 잠금은 데이터 공유 환경에서만 사용됩니다. 반의어 - *논리 잠금(L-잠금)*.

물리적 청구. OS/390용 DB2 UDB에서, 전체 비파티션 색인에 대한 청구.

물리적으로 완료됨. OS/390용 DB2 UDB에서, 동시 복사 프로세스가 완료되고 출력 데이터 세트가 작성된 상태.

미구분 ASCII(ASC) 형식. 데이터 가져오기에 사용되는 파일 형식. 미구분 ASCII는 ASCII 제품과의 데이터 교환에 사용되는 행 분리 문자가 있는 순차 ASCII 파일입니다.

미압축 속성. 테이블에 현재 데이터가 아닌, 데이터 변경 실행기록이 들어 있음을 나타내는 테이블 속성. 이 속성 세트가 있는 테이블에는 각 키 값에 대해 한 행 이상이 포함됩니다.

미입력 매개변수 표시문자. 목표 데이터 유형 없이 지정되는 매개변수 표시문자. 작은 따옴표 형태를 지닙니다.

미협력 트랜잭션. 여러 자원에 액세스하지만, 호가약 및 롤백이 트랜잭션 관리 프로그램에 의해 조정되지 않는 트랜잭션.

미확약 읽기(UR). 응용프로그램이 다른 트랜잭션의 미확약 변경사항에 액세스할 수 있도록 하는 분리 레벨. 다른 응용프로그램이 테이블을 삭제하거나 교체하려 하지 않는 한, 응용프로그램은 행으로부터 다른 응용프로그램을 잠그지 않습니다.

바

바이트 리버설. 수치 데이터 중에서 가장 덜 중요한 바이트가 우선 저장되는 기술.

바인더리 오브젝트 이름. NetWare 파일 서버에서 바인더리 오브젝트 이름을 포함하는 48 바이트 문자열. 데이터베이스 관리 프로그램 구성 필드, 오브젝트 이름은 DB2 서버 인스턴스를 고유하게 나타내며, NetWare 파일 서버에 있는 바인더리에 오브젝트로서 저장됩니다.

바인드. (1) SQL에서, SQL 사전 처리 컴파일러의 출력이 *액세스 플랜*이라고 하는 사용 가능한 구조로 변환되는 프로세스. 이 프로세스 중에, 데이터에 대한 액세스 경로가 선택되고 일부 권한 부여 점검이 수행됩니다. (2) OS/390용 DB2 UDB에서, DBMS 사전 처리 컴파일러의 출력이 사용 가능한 제어 구조(*패키지* 또는 *응용프로그램 플랜*이라고 함)로 변환되는 프로세스. 프로세스 중에, 데이터에 대한 액세스 경로가 선택되고 일부 권한 부여 점검이 수행됩니다. 또한 *자동 리바인드*, *동적 바인드*, *점증 바인드*, *정적 바인드*도 참조.

바인드 파일. **bind** 명령 또는 API가 BINDFILE 옵션과 함께 사용될 때 사전 처리 컴파일러에 의해 산출되는 파일. 이 파일에는 응용프로그램에 있는 모든 SQL문에 대한 정보가 포함됩니다.

반복 읽기(RR). 트랜잭션 내에서 참조되는 응용프로그램의 모든 행을 잠그는 분리 레벨. 프로그램이 반복 읽기(RR) 보호를 사용하는 경우, 프로그램이 현재 트랜잭션을 종료할 때까지 프로그램에 의해 참조되는 행들이 다른 프로그램에 의해 변경될 수 없습니다.

방향 지정된 조인. 조인 테이블 둘다 또는 둘 중 하나가 조인 술어에 기초한 새로운 데이터베이스 파티션으로 개작되어 방향 지정되는 모든 행의 관계 연산. 테이블의 모든 파티션 키 컬럼이 equijoin 술어에 참여하는 경우, 다른 테이블은 개작됩니다. 그렇지 않으면(적어도 하나의 equijoin 술어가 있는 경우) 두 테이블 모두 개작됩니다.

배정밀도 부동 소수점 수. SQL에서, 실수에 대한 대략적인 64 비트 표시.

백업 보류. 데이터베이스나 테이블 공간이 백업될 때까지 조작이 수행되지 못하도록 하는 데이터베이스 또는 테이블 공간의 상태.

백워드 로그 복구. OS/390용 DB2 UDB가 중단된 모든 변경사항에 대해 UNDO 로그 레코드를 적용하기 위해 역방향으로 로그를 검색하는 재시작 프로세스의 네 번째 및 최종 단계.

버전. OS/390용 DB2 UDB에서, 유사한 프로그램, DBRMs, 패키지 또는 LOB 세트의 구성원.

- 프로그램 버전은 프로그램을 사전에 컴파일하여 생성되는 소스 코드입니다. 프로그램 버전은 프로그램 이름과 시간소인(일관성 토큰)에 의해 식별됩니다.
- DBRM의 버전은 프로그램을 사전에 컴파일하여 생성되는 DBRM입니다. DBRM 버전은 해당되는 프로그램 버전과 같은 프로그램 이름과 시간소인에 의해 식별됩니다.
- 패키지 버전은 특정 데이터베이스 시스템 내에서 DBRM을 바인딩한 결과입니다. 패키지 버전은 DBRM과 같은 프로그램 이름과 일관성 토큰으로 식별됩니다.
- LOB 버전은 특정 시점에서 LOB 값의 사본입니다. LOB에 대한 버전 번호는 LOB에 대한 보조 색인 항목에 저장됩니다.

버퍼 풀. OS/390용 DB2 UDB에서, 하나 이상의 테이블 공간이나 색인에 대한 버퍼링 요구사항을 만족시키기 위해 예약되어 있는 주 저장영역.

변경 누적 테이블. DB2 복제에서, 소스 테이블에 대해 기록된 변경사항을 기초로 하는 데이터 집계를 포함하는 목표 테이블의 유형.

변경 데이터(CD) 테이블. 복제 소스 테이블에 대해 변경된 데이터가 들어 있는 소스 서버에서의 복제 제어 테이블.

변수. 변경할 수 있는 값을 지정하는 데이터 요소.

변환. Data Warehouse Center에서, 데이터에 대해 수행되는 조작. 피벗 및 정리는 변환의 유형입니다.

변환기. 웨어하우스 데이터에서 작동되는 프로그램. Data Warehouse Center는 두 가징 유형의 변환기인 통계 변환기와 웨어하우스 변환기를 제공합니다. 통계 변환기는 하나 이상의 테이블에 있는 데이터에 대한 통계를 제공하고, 웨어하우스 변환기는 분석을 위한 데이터를 제공합니다. 각 단계에는 데이터 조작 유형을 수행하는 프로세스에서 사용되는 변환기에 해당되는 유형이 있습니다. 예를 들어, 정리 단계에서는 정리 변환기를 사용합니다.

별명. 테이블, 뷰, 데이터베이스 또는 별명을 식별하기 위해 사용되는 대체 이름. 별명은 같은 DB2 서브시스템이나 원격 DB2 하위 시스템의 테이블이나 뷰를 참조하기 위해 SQL문에서 사용할 수 있습니다.

별명 체인. 서로 순차적이면서 비반복적인 형태로 참조하는 일련의 테이블 별명들.

별명(nickname). (1) 데이터 소스 테이블이나 뷰를 참조하기 위해 연합 서버에서 사용하는 식별자. (2) IBM 데이터베이스가 아닌 다른 데이터베이스에서 물리 데이터베이스 오브젝트(예: 테이블 또는 저장 프로시듀어)를 나타내기 위해 DB2 DataJoiner 데이터베이스에서 정의되는 이름.

병렬 그룹. OS/390 환경에서, 병렬로 실행하고 병렬 타스크 수가 같은 연속 조작 세트.

병렬 세션. SNA에서, 같은 두 개의 논리 장치 사이에 있는 현재 활동중인 두 개 이상의 세션. 각 세션에는 서로 다른 세션 매개변수가 있을 수 있습니다. *세션* 참조.

병렬 처리. 동시에(병렬로) 여러 개의 데이터베이스 연산을 수행하는 기능. *파티션간 병렬 처리*, *파티션 내 병렬 처리*, *병렬 I/O*에서 참조하십시오.

병렬 타스크. OS/390 환경에서, 조회를 병렬로 처리하기 위해 동적으로 작성되는 실행 단위. 이것은 MVS 서비스 요청 블록에 의해 구현됩니다.

병렬 I/O. 응답 시간을 줄이기 위해 동시에 두 개 이상의 I/O 장치로부터 읽거나 기록하는 프로세스.

병렬 I/O 처리. OS/390용 DB2 UDB가 단일 사용자 조회에 대한 여러 개의 동시 요청을 초기화하고 여러 데이터 파티션에서 동시에(병렬로) I/O 처리를 수행합니다.

병렬 Sysplex. 특정의 멀티시스템 하드웨어 구성요소 및 소프트웨어 서비스를 통해 서로 통신하고 협동하는 OS/390 시스템 세트.

병렬화 정도. OS/390용 DB2 UDB에서, 조회를 처리하기 위해 초기화되는, 동시에 실행되는 조작 수.

보유 잠금. 서브시스템 실패 시 OS/390용 DB2 UDB 서브시스템이 보유하고 있었던 MODIFY 잠금. 잠금은 OS/390용 DB2 UDB 실패에서 결합 기능 잠금 구조에 보유됩니다.

보조 색인. OS/390용 DB2 UDB에서, 각 색인 항목이 LOB를 참조하는 보조 테이블의 색인.

보조 테이블. OS/390용 DB2 UDB에서, 컬럼들이 정의되어 있는 테이블 밖에 컬럼을 정의하는 테이블. 반의어 - *기본 테이블*.

보호 대화. OS/390 환경에서, 2단계 확약 흐름을 지원하는 VTAM 대화.

복구. (1) 손상 후 작동 가능 상태로 시스템 또는 시스템에 저장된 데이터를 재설정하는 것. (2) 백업을 복원하고 이와 연관된 로그를 롤 포워드함으로써 데이터베이스를 재구축하는 과정.

복구 단위. OS/390용 DB2 UDB 인스턴스와 같이, 단일 자원 관리 프로그램 내에서 복구할 수 있는 일련의 조작. 반의어 - *작업 단위(UOW)*.

복구 로그. *데이터베이스 로그* 참조.

복구 보류. 데이터베이스 또는 테이블 공간 상태. 데이터베이스 또는 테이블 공간은, 백업으로부터 복원될 때 복구 보류 상태에 있습니다. 데이터베이스나 테이블 공간이 이 상태에 있는 동안에는 데이터에 액세스할 수 없습니다.

복구 연기됨 취소 UR. OS/390용 DB2 UDB에서, 1단계 확약중 실패하거나 구간 복원중 실패했고, 시스템 장애나 취소에 의해 인터럽트되었으며, 재시작 중에 백아웃을 완료하지 못한 복구 단위.

복구 토큰. OS/390용 DB2 UDB에서, 복구 시 사용되는 요소에 대한 식별자(예: *NID* 또는 *URID*).

복구가능 로그. 모든 로그 레코드가 보유되어 있는 데이터베이스 로그. 이 로그로 인해, 시스템 실패가 발생할 경우, 소실된 데이터는 포워드 복구로 복구할 수 있습니다. 반의어 - *순환 로그*.

복사 작업 내역. *복사 작업 내역 세트* 참조.

복사 작업 내역 세트. DB2 복제에서, 소스 테이블, 목표 테이블 및 변경된 데이터의 복제를 다루는 제어 정보의 스펙. 또한 **복사 작업 내역 세트 구성원** 참조.

복사 작업 내역 세트 구성원. DB2 복제에서, 복사 작업 내역 세트의 구성원. 각 소스-목표 쌍마다 하나의 구성원이 있습니다. 각 구성원은 목표 테이블의 구조와, 소스 테이블에서 복제될 행과 컬럼을 정의합니다.

복사 작업 내역 순환. DB2 복제에서, Apply 프로그램이 주어진 복사 작업 내역 세트에 대해 변경된 데이터를 검색하고, 변경사항을 목표 테이블에 복제하며, 적절한 복제 제어 테이블을 갱신하여 수행된 진행을 반영하는 프로세스.

복사 작업 내역 프로세스. DB2 복제에서, 복사 작업 내역 세트와 복사 작업 내역 세트 구성원을 정의하는 프로세스. 반의어 - **등록 프로세스**.

복수 바이트 문자 세트(MBCS). 각 문자가 2 바이트 이상으로 표시되는 문자 세트. 2 바이트만 사용하는 문자 세트는 일반적으로 *2 바이트 문자 세트*로 더 알려져 있습니다.

복원. 사용중인 저장영역 위치로 백업 사본을 되돌리는 것.

복원 세트. 복원 및 롤 포워드될 때 데이터베이스나 테이블공간을 다시 일관된 상태로 되돌리는 제로 또는 그 이상의 로그 파일과 데이터베이스 또는 테이블 공간의 백업 사본.

복제. 여러 위치에서 정의된 데이터 세트를 유지보수하는 프로세스. 여기에는 한 위치(소스)에 대한 지시된 변경사항을 또 다른 위치(목표)로 복사하고 그 데이터를 두 위치에서 동기화하는 작업이 포함됩니다.

복제. 복사 작업 내역 정의를 통해 사용자 테이블로부터의 갱신을 받고 지역적으로 갱신될 수 있는 목표 테이블 유형. 이는 사용자 테이블 또는 읽기 전용 목표 테이블을 갱신하기 위한 소스가 될 수 있습니다.

복제 관리자. 복제 소스 및 복사 작업 내역을 정의하는 담당자. 이 사용자는 Capture 및 Apply 프로그램도 수행할 수 있습니다.

복제 목표 테이블. 목표 서버에서 update-anywhere 목표 테이블의 유형인 복제 테이블.

복제 복사 작업 내역. 데이터 보강 옵션과 함께, 지정된 시간 및 빈도로 복제 소스로부터 목표 테이블로 변경된 데이터를 복사하기 위한 스펙. Apply 프로그램이 데이터를 복사하는 데 필요한 모든 정보를 정의합니다.

복제 소스. 복사 요청을 승인할 수 있고, 복사 작업 내역 세트의 소스 테이블이기도 하느 데이터베이스 테이블 또는 뷰. 또한 *복사 작업 내역 세트* 참조.

복합 키. 같은 테이블의 순서화된 키 컬럼 세트.

복합 SQL문. 응용프로그램 서버(AS)로의 단일 호출에서 실행되는 SQL문 블록.

부분 새로 고침. DB2 복제에서, 기존 데이터를 대체하면서, 변경된 데이터만 목표 테이블로 복사하는 프로세스. 반의어 - **전체 새로 고침**.

부속 구성요소. 일반적인 함수를 제공하기 위해 함께 작동되는, 거의 관련되는 OS/390용 DB2 UDB 모듈들의 그룹.

부속 선택. ORDER BY절, UPDATE절 또는 UNION 연산자를 포함하지 않는 조회 양식.

부속 에이전트. 부속 요청에서 작동되는 에이전트 유형. 단일 응용프로그램이 여러 번의 요청을 할 수 있고, 각 요청은 여러 개의 부속 요청으로 나뉠 수 있습니다. 따라서, 동일한 응용프로그램대신 여러 개의 부속 에이전트가 작동될 수 있습니다. 응용프로그램에 대해 작동 중인 모든 부속 에이전트는 그 응용프로그램에 대한 조정 에이전트에 의해 조정됩니다.

부속 조회. 다른 SQL문의 WHERE 또는 HAVING 절 내에 있는 SELECT문. 중첩되는 SQL문.

부트스트랩 데이터 세트(BSDS). OS/390용 DB2 UDB에 대한 이름 및 상태 정보와, 모든 활동중 로그 데이터 세트 및 아카이브 로그 데이터 세트에 대한 RBA 범위 스펙이 있는 VSAM 데이터 세트. OS/390용 DB2 UDB 디렉토리 및 카탈로그에 대한 암호와 조건부 재시작 및 체크포인트 레코드 목록도 포함되어 있습니다.

분리 레벨. 동시에 실행되는 다른 응용프로그램 프로세스와 응용프로그램 프로세스가 고립되어 있는 정도를 정의하는 속성.

분리 문자. 데이터 항목을 그룹화하거나 분리시키는 플래그 또는 문자.

분리 문자 토큰. 구문 도표에 표시된 특수 문자, 연산자 기호, 분리 식별자 또는 리터럴.

분리 식별자. 큰 따옴표로 묶어 있는 일련의 문자들. 이 연속은 글자 하나와 그 뒤에 제로 또는 여러 문자로 구성되며, 각 문자는 글자, 숫자 또는 밑줄 문자가 될 수 있습니다.

분리(fenced). DBMS가 함수에 의해 수정되지 못하도록 보호하기 위해 정의된 사용자 정의 함수(UDF) 또는 저장 프로시듀어의 유형에 관련된 용어. DBMS는 장벽에 의해 함수나 저장 프로시듀어로부터 고립되어 있습니다. 반의어 - *비분리*.

분산 관계형 데이터베이스. 서로 다르지만 상호 연결된 전산 시스템상에 테이블이 저장되어 있는 데이터베이스.

분산 네트워크 디렉토리. *분산 디렉토리 데이터베이스*.

분산 데이터 기능(DDF). OS/390용 DB2 UDB가 다른 RDBMS와 통신할 때 거치는 OS/390용 DB2 UDB 구성요소 세트.

분산 디렉토리 데이터베이스. APPN 네트워크를 통해 흩어져 있는 개별 디렉토리에 유지보수되는 네트워크내 모든 자원의 전체 목록. 각 노드에는 완전한 디렉토리 조각이 있으나, 한 노드에 전체 목록이 있을 필요는 없습니다. 시스템 정의, 조작자 조치, 자동 등록, 진행중인 네트워크 검색 프로시듀어를 통해 항목들이 작성, 수정 및 삭제됩니다. 동의어 - *분산 네트워크 디렉토리*.

분산 요청(DR). 연합 데이터베이스 시스템에서, 두 개 이상의 데이터 소스로 보내지는 SQL 조회.

분산 작업 단위(DUOW). SQL문이 SQL문 하나당 하나의 시스템에 한해 여러 관계형 데이터베이스 관리 시스템으로 제출될 수 있도록 하는 작업 단위(UOW).

불명확한 커서. (1) 정의 또는 문맥으로부터 갱신 가능하거나 읽기 전용인 것으로 판별될 수 없는 커서. (2) OS/390용 DB2 UDB에서, FOR FETCH ONLY 절이나 FOR UPDATE OF 절에서 정의되지 않은 데이터베이스 커서는 읽기 전용 결과 테이블에도 정의되지 않고, SQL UPDATE 또는 DELETE 명령문에 있는 WHERE CURRENT 절의 목표도 아니며, PREPARE 또는 EXECUTE IMMEDIATE SQL 명령문을 포함하는 플랜이나 패키지에 있습니다.

불변 문자 세트. OS/390용 DB2 UDB에서, (1) 코드 포인트 지정사항이 코드 페이지간에 변경되지 않는 구문 문자 세트와 같은 문자 세트. (2) 모든 문자 세트의 부분으로 사용할 수 있는 최소 문자 수.

불변 함수. 그 결과가 입력 인수 값에 의해서만 영향받는 사용자 정의 함수. 동일한 인수 값을 가진 연속 호출은 항상 동일한 결과를 산출합니다. 반의어 - *상이 함수*.

뷰. 조회에 의해 생성되는 데이터로 구성되는 논리 테이블. 반의어 - *기본 테이블*.

뷰 점검 옵션. OS/390용 DB2 UDB에서, 뷰를 통해 삽입되거나 갱신되는 모든 행이 그 뷰의 정의를 따라야 하는 지를 지정하는 옵션. 뷰 점검 옵션은 CREATE VIEW문의 WITH CASCADED CHECK OPTION, WITH CHECK OPTION 또는 WITH LOCAL CHECK OPTION 절에서 지정할 수 있습니다.

브라우저. 컴퓨터 모니터에서 텍스트로 표시 가능한 Text Extender 함수.

브로드캐스트 조인. 테이블의 모든 파티션이 모든 노드로 전송되는 조인.

블로킹. 응용프로그램을 바인딩할 때 지정되는 옵션. 통신 서브시스템이 여러 행의 정보를 캐쉬할 수 있도록 함으로써, 네트워크를 통해 각 요청에 대해 각 FETCH문이 한 행을 전송할 필요가 없습니다. 반의어 - *데이터 블로킹*.

블록. 하나의 단위로서 기록되거나 전송되는 데이터 요소의 문자열.

비 리프 페이지. OS/390용 DB2 UDB에서, 색인에 다른 페이지(리프 또는 비 리프 페이지)의 키와 페이지 번호를 포함하는 페이지. 비 리프 페이지는 실제 데이터를 지시하지 않습니다. 반의어 - *리프 페이지*.

비결정 함수. OS/390용 DB2 UDB에서, 결과가 오로지 입력 인수의 값에만 종속되지는 사용자 정의 함수. 동일한 인수 값을 사용한 연속 호출에서 다른 응답을 받을 수 있습니다. 이러한 유형의 함수를 *가변* 함수라고도 합니다. 반의어 - *결정 함수*(항상 같은 입력에 대해 같은 결과를 생성하므로 *불변 함수*라고도 함).

비교 연산자. 비교 표현식에서 사용되는 접두어 연산자. 비교 연산자는 ¬<(보다 작지 않음), <=(이하), ¬=(같지 않음), =(같음), >=(이상), >(초과), ¬>(보다 크지 않음)입니다.

비동기. 시간이 정기적이 아니므로, 프로그램 지시 처리를 예측이나 예상하지 못하는 것. 반의어 - *동기*.

비동기 일괄처리 갱신. 소스의 모든 변경사항이 기록되고 지정된 간격으로 기존의 목표 데이터에 적용되는 프로세스. 반의어 - *비동기 지속 갱신*.

비동기 지속 갱신. 기본 테이블에 확약된 후 소스의 모든 변경사항이 기존의 목표 데이터에 기록되거나 적용되는 프로세스. 반의어 - *비동기 일괄처리 갱신*.

비분리. DBMS 프로세스에서 수행되도록 정의된 저장 프로시듀어 또는 사용자정의 함수 유형. 반의어 - *분리(fenced)*.

비용 범주. OS/390용 DB2 UDB가 명령문이 바인드될 때 SQL문의 비용 측정을 하는 범주. 비용 예측은 다음의 비용 범주 중 하나에서 이루어집니다.

- A: OS/390용 DB2 UDB가 기본값을 사용하지 않고 비용 예측을 할 수 있을만큼 충분한 정보를 가지고 있음을 나타냅니다.
- B: 예측을 위해 OS/390용 DB2 UDB가 기본값을 사용하도록 하는 상태가 있음을 나타냅니다.

비용 범주는 명령문이 Explain될 때 DSN_STATEMNT_TABLE의 COST_CATEGORY 컬럼에서 외부로 처리됩니다.

비정상 종료. (1) 작업이 비정상적으로 종료되도록 하는 조작자의 조치 또는 시스템 실패. (2) DB2에서, trap 또는 segv와 같은, 프로그램 제어 하에 있지 않은 exit.

비지니스 메타데이터. 비지니스 용어로 정보 자산을 설명하는 데이터. 비지니스 메타데이터는 정보 카탈로그에 저장되며 사용자들이 필요로 하는 정보를 찾거나 이해하기 위해 액세스합니다. 예를 들어, 프로그램의 비지니스 메타데이터에는 그 프로그램이 무엇이고 사용하는 테이블은 무엇인지에 대한 설명이 있습니다. 반의어 - *기술적 메타데이터*.

비지니스 이름. Data Warehouse Center에서, 단계를 언급하는 이름. 각 단계에는 연관되는 비지니스 이름과 DB2 테이블 이름이 있습니다. 비지니스 이름은 보통 웨어하우스 사용자가 사용하며, DB2 테이블 이름은 SQL문에서 사용됩니다.

비트 데이터. 코드화된 문자 세트와 연관되지 않아서 결코 변환되지 않는 CHAR 또는 VARCHAR 문자 유형의 데이터.

비파티션 색인. OS/390용 DB2 UDB에서, 파티션 색인이 아닌 색인.

사

사실 테이블. OLAP Starter 킷에서의 테이블, 또는 대부분의 경우에서 관계형 큐브에 대한 모든 데이터 값을 포함하는 DB2에 있는 네 개의 테이블 세트.

사용. Text Extender에서 사용할 수 있도록 데이터베이스, 텍스트 테이블 또는 텍스트 컬럼을 준비하는 것.

사용안함. 사용 가능화 프로세스 동안 작성된 항목을 제거하여 Text Extender에 대해 사용 가능하게 되기 전에 데이터베이스, 텍스트 테이블 또는 텍스트 컬럼을 해당되는 원래 상태로 복원하는 것.

사용자 맵핑. 사용자가 연합 서버에 연결하는 권한과 사용자가 데이터 소스에 연결하는 권한 사이의 연관.

사용자 복사 테이블. DB2 복제에서, 내용이 모든 소스 테이블이나 소스 테이블의 일부와 일치하고 사용자 데이터 컬럼만 포함하는 목표 테이블.

사용자 정의 구별 유형. *구별 유형* 참조.

사용자 정의 데이터 유형(UDT). *구별 유형* 참조.

사용자 정의 성능 변수. 사용자에 의해 작성되어 성능 변수 프로파일에 추가되는 성능 변수.

사용자 정의 유형(UDT). 사용자에 의해 작성되었고 데이터베이스 관리 프로그램에 원시인 것이 아닌 데이터 유형. OS/390용 DB2 UDB에서, *구별 유형*은 사용자 정의 유형 대신 사용됩니다.

사용자 정의 프로그램. Data Warehouse Center와 함께 포함되어 자동으로 정의되는 제공된 프로그램들과 반대로, 사용자가 제공하고 Data Warehouse Center에 대해 정의하는 프로그램.

사용자 정의 함수(UDF). 데이터베이스 관리 시스템에 대해 정의되어 SQL 조회에서 참조될 수 있는 함수. 다음과 같은 함수가 가능합니다.

- 외부 함수. 호출될 때마다 스칼라 결과가 산출되고 인수가 스칼라 값인 프로그래밍 언어로 함수 내용이 작성됩니다.
- 전래 함수. 이미 DBMS에 알려진 사용자 정의 함수 또는 또다른 내장 함수에 의해 구현됩니다. 이 함수는 스칼라 함수나 컬럼(총계) 함수이고, 값 집합으로부터 단일 값을 리턴합니다(예, MAX, AVG).

사용자 테이블. DB2 복제에서, 복제 소스로 정의되기 전에 응용프로그램에 대해 작성되어 그 응용프로그램에서 사용되는 테이블. 이것은 읽기 전용 목표 테이블, 데이터 일관 변경 테이블 및 행 Replica 테이블에 대한 갱신사항의 소스로 사용됩니다.

사용중인 로그. (1) DB2 UDB에서, 복구 및 구간 복원을 위해 현재 필요한 1차 및 2차 로그 파일. 반의어 - *아카이브 로그*. (2) 로그 레코드가 생성되는 대로 기록되는 OS/390용 DB2 UDB 로그의 부분. 사용중인 로그에는 항상 최근 로그 레코드가 있는 반면, 아카이브 로그에는 오래되어서 더이상 사용중인 로그에 맞지 않는 레코드가 보유됩니다.

사인온. OS/390용 DB2 UDB 자원을 사용할 수 있는 권한이 부여되어 있는지 확인하기 위해 OS/390용 DB2 UDB를 사용하기 위해 접속 기능에 의해 개별적 CICS 또는 IMS 응용프로그램 프로세스 대신 수행되는 요청.

사전. Text Extender가 특정 언어로된 문서를 분석, 색인화, 검색 및 강조표시하는 동안 사용하는 언어 관련 정보의 콜렉션.

사전 이미지. DB2 복제에서, 데이터 변경 테이블이나 데이터베이스 로그 또는 저널에 기록된, 새로 고치기 이전의 소스 테이블 컬럼 내용. 반의어 - *사후 이미지*.

사전 처리 컴파일. SQL문이 컴파일되기 전에 그 SQL문을 포함하는 프로그램을 처리하는 것. SQL문은 호스트 언어 컴파일러에 의해 인식될 명령문으로 대체됩니다. 사전 처리 컴파일 프로세스로부터의 출력에는 컴파일러에 제출되고 바인드 프로세스에서 사용될 수 있는 소스 코드가 포함됩니다.

사후 이미지. DB2 복제에서, 데이터 변경 테이블이나 데이터베이스 로그 또는 저널에 기록된 소스 테이블 요소의 갱신된 내용. 반의어 - *사전 이미지*.

삭제 규칙. 상위 행의 삭제를 제한하거나 종속 행에서의 그러한 삭제 효과를 지정하는 참조 제한조건과 연관된 규칙.

삽입 규칙. 행을 테이블에 삽입하기 위해 충족시켜야 하는 데이터베이스 관리 프로그램의 조건.

삽입 트리거. OS/390용 DB2 UDB에서, 트리거링 SQL 조작 INSERT로 정의되는 트리거.

상관 부속 조회. 부속 조회 외부에 있는 테이블의 컬럼에 대한 상관 참조를 포함하는 부속 조회.

상관 이름. 단일 SQL문 내부에 있는 테이블 또는 뷰를 지정하는 식별자. 이는 FROM절 또는 UPDATE 및 DELETE 문의 첫번째 절에 정의될 수 있습니다.

상관 참조. 부속 조회 외부에 있는 테이블의 컬럼에 대한 참조.

상관 컬럼. SQL에서, 하나의 컬럼값과 다른 컬럼값 사이의 관계.

상관 ID. OS/390용 DB2 UDB에서 특정 스레드와 연관된 식별자. TSO에서, 이는 권한 부여 ID나 작업 이름입니다.

상대 논리 장치(LU). (1) SNA에서, 세션에 참가하는 원격 구성원. (2) VTAM 대화에 의해 지역 OS/390용 DB2 UDB 서브시스템에 연결되는 SNA 네트워크의 액세스 지점.

상수. 변경되지 않는 값을 지정하는 언어 요소. 상수는 문자열이나 상수로 구분됩니다. 반의어 - *변수*.

상위 잠금. OS/390용 DB2 UDB에서의 명시적 계층 구조 잠금의 경우, 계층 구조에서 아래 부분에 있는 하위 잠금을 가지고 있는 자원에서 보유되는 잠금. 보통, 테이블 공간이나 파티션 지향 잠금은 상위 잠금입니다.

상위 키. 참조 제한조건에 사용되는 고유 키 또는 기본 키. 상위 키 값은 제한조건에 있는 외부 키의 유효 값을 결정합니다.

상위 테이블. 적어도 하나의 참조 제한조건이 있는 상위인 테이블.

상위 테이블 공간. OS/390용 DB2 UDB에서, 상위 테이블을 포함하는 테이블 공간. 그 테이블의 종속 요소를 포함하는 테이블 공간은 종속 테이블 공간입니다.

상위 행. 종속 행이 적어도 하나 이상 있는 행.

상태. Data Warehouse Center에서, 스케줄됨, 데이터 처리 중 또는 성공과 같은, 한 단계의 진행 중인 작업 처리 조건.

상호 메모리 링크. OS/390 환경에서, 서로 다른 주소 공간에서 프로그램을 호출하기 위한 방법. 호출은 호출자에 대해 동시에 발생합니다.

색인. 키의 값들에 의해 논리적으로 순서화되는 포인터들의 집합. 색인은 데이터에 대한 신속한 액세스를 제공하고, 테이블내 행에 대한 고유성을 향상시킬 수 있습니다.

색인 공간. OS/390용 DB2 UDB에서, 하나의 색인에 대한 항목들을 저장하기 위해 사용되는 페이지 세트.

색인 스펙. 연합 데이터베이스 시스템에서, 데이터 소스 테이블에 관련되는 메타데이터 세트. 이 메타데이터는 보통 색인 정의에 포함되는 정보(예: 정보를 신속하게 검색하기 위해 검색할 컬럼)로 구성됩니다. 사용자는 테이블에 색인이 전혀 없거나 연합 서버에 알려져 있지 않은 색인이 있는 경우 이 메타데이터를 연합 서버에 제공할 수도 있습니다. 메타데이터의 목적은 테이블의 데이터를 검색하는 것입니다.

색인 키. 색인 항목 순서를 결정하는 데 사용되는 테이블의 컬럼 세트.

색인 파일. Video Extender에서 컷이나 비디오 클립의 개별적 프레임을 찾기 위해 사용되는 색인화 정보가 있는 파일.

색인 파티션. 해당 노드에서 테이블 파티션과 연관된 색인의 일부. 테이블에 정의된 색인은 여러 개의 색인 파티션에 의해 테이블 파티션당 하나씩 구현됩니다.

색인 sargable 술어. 규정 요청을 규정하는 색인 항목 수를 줄이기 위해 색인 리프 페이지의 색인 항목에 적용되는 SQL 술어. 이러한 술어는 액세스되는 데이터 행 수를 줄이는 데 도움을 줍니다.

서버. (1) 네트워크에서, 다른 스테이션(예: 파일 서버, 프린터 서버, 메일 서버)에 기능을 제공하는 노드. (2) 연합 데이터베이스 시스템에서, 연합 서버에 대해 데이터 소스를 식별하는 정보 단위. 이 정보에는 서버의 이름, 해당 유형, 해당 버전, 그리고 연합 서버가 데이터 소스로부터 데이터를 수신하고 통신하기 위해 사용하는 랩퍼의 이름이 포함될 수 있습니다. (3) 네트워크를 통해 하나 이상의 클라이언트에 서비스를 제공하는 기능적 단위. OS/390용 DB2 UDB 환경에서, 서버는 원격 RDBMS로부터의 요청에 대한 목표이고 데이터를 제공하는 RDBMS입니다. 또한 *응용프로그램 서버* 참조.

서브시스템. OS/390용 DB2 UDB에서, 관계형 데이터베이스 관리 시스템(RDBMS)의 구별 인스턴스.

서브페이지. OS/390용 DB2 UDB에서, 실제 색인 페이지가 나눠질 수 있는 단위.

서비스 이름. 포트 번호가 원격 노드에서 사용되도록 지정하는 기호 방법을 제공하는 이름. TCP/IP 연결에는 원격 노드의 주소와, 응용프로그램을 식별하기 위해 원격 노드에서 사용될 포트 번호가 필요합니다.

서비스 클래스. OS/390용 DB2 UDB에서, 고객 성능 목표를 특정의 DDF 스레드나 저장 프로시듀어와 연관시키기 위해 MVS 워크로드 관리 프로그램에서 사용되는 8자 식별자. 서비스 클래스는 병렬 보조에 대한 작업을 분류하기 위해서도 사용됩니다.

서비스 클래스. OS/390용 DB2 UDB에서, 네트워크를 거치는 라우트 목록에 대한 VTAM 용어로, 사용 우선 순위대로 정렬됩니다.

선언 생성기(DCLGEN). 테이블에 따라 SQL 테이블 선언과 COBOL, C 또는 PL/I 데이터 구조 선언을 생성하는 OS/390용 DB2 UDB의 부속 구성요소. 선언은 OS/390용 DB2 UDB 시스템 카탈로그 정보에서 생성됩니다. DCLGEN은 DSN 부속 명령이기도 합니다.

선형 데이터 세트(LDS). OS/390 환경에서, 데이터는 포함하지만 제어 정보를 포함하지 않는 VSAM 데이터 세트. 선형 데이터 세트는 가상 저장영역에서 바이트 주소 지정 가능한 문자열로 액세스할 수 있습니다.

설명. SQL 컴파일러가 SQL문을 해석하기 위해 선택한 액세스 플랜에 대한 상세한 정보를 얻는 것. 이 정보는 액세스 플랜을 선택하는 데 사용되는 결정 기준을 기술합니다.

설명 가능 명령문. 설명 조작이 수행될 수 있는 SQL문. 설명 가능한 명령문은 SELECT, UPDATE, INSERT, DELETE, VALUES입니다.

설명된 명령문. 설명 조작이 수행된 SQL문.

설치. OS/390 서브시스템으로 작동되도록 OS/390용 DB2 UDB 서브시스템을 준비하는 프로세스.

설치 검증 시나리오. 주 OS/390용 DB2 UDB 함수를 실습하고 OS/390용 DB2 UDB가 올바르게 설치되었는지 테스트하는 일련의 조작.

성능 모니터. 조정을 목적으로 데이터베이스 관리자가 그래픽 인터페이스를 사용하여 DB2 시스템 성능을 모니터하도록 하는 도구. 이 도구는 제어 센터에 의해 액세스될 수 있습니다.

성능 변수. 데이터베이스 관리 프로그램의 성능 데이터로부터 도출되는 통계. 이 변수에 대한 표현식은 사용자가 정의할 수 있습니다.

성능 변수 프로파일. 성능 변수 정의가 포함된 플랫 파일. 이 파일은 편집, 복사, 공유가 가능합니다. 동일한 성능 모니터이 서로 다른 프로파일을 사용할 수 있으므로 다른 계산이 수행될 수 있습니다.

성능 스냅샷. 일정 시점에서 데이터베이스 관리 프로그램으로부터 검색되는 데이터베이스 오브젝트 세트에 대한 성능 데이터.

성능 행렬. 동일한 데이터베이스 오브젝트에 속해 있는 모든 성능 변수의 콜렉션.

세 부분 이름. 테이블, 뷰 또는 별명의 전체 이름. 위치 이름, 권한 부여 ID 및 오브젝트 이름으로 구성되고, 마침표로 구분됩니다.

세계 표준시(UTC). 동의어 - 그리니지 표준시.

세그먼트된 테이블 공간. OS/390용 DB2 UDB에서, 세그먼트라고 하는 같은 크기의 페이지 그룹으로 나눠지는 테이블 공간. 세그먼트들은 서로 다른 테이블의 행들이 같은 세그먼트에 저장되지 않도록 테이블에 지정됩니다.

세부 함수 이름. (1) 시스템에 함수를 고유하게 식별시키는 이름. (2) OS/390용 DB2 UDB에서, 특정 이름으로 데이터베이스 관리 프로그램에 알려져 있는 특정의 사용자 정의 함수. 많은 특정의 사용자 정의 함수가 같은 함수 이름을 가질 수 있습니다. 데이터베이스에 대해 사용자 정의 함수(UDF)가 정의될 경우, 모든 함수마다 해당되는 스키마 내에서 고유한 특정 이름이 지정됩니다. 사용자가 이 이름을 제공할 수 있으며, 그렇지 않으면 기본값이 사용됩니다.

세션. 두 개의 스테이션 또는 NAU가 통신할 수 있도록 하는 두 스테이션 또는 SNA 네트워크 주소지정 가능 장치(NAU) 사이의 논리적 연결.

세션 바인드해제(UNBIND). 두 논리 장치(LU) 사이의 세션을 비활성화하는 요청.

세션 보안. LU 6.2인 경우, 상대 LU 검증 및 세션 데이터 암호화. 데이터가 암호화 형식으로 전송되도록 하는 SNA(Systems Network Architecture) 함수.

세션 상대. SNA에서, 사용 중인 세션에 참여하고 있는 두 네트워크 주소지정 가능 장치(NAU) 중 하나.

세션 제한. SNA에서, 특정의 논리 장치(LU)에서 지원할 수 있는, 현재 활동중인 최대 논리 장치간(LU간) 세션 수.

세션 프로토콜. OS/390용 DB2 UDB에서, 사용 가능한 SNA 통신 요청 및 응답 세트.

세트 연산자. 관계 연산자 합집합, 차집합, 교집합에 해당하는 SQL 연산자 UNION, EXCEPT, INTERSECT. 집합 연산자는 두 개의 서로 다른 결과 테이블을 결합하여 결과 테이블 하나를 도출합니다.

섹션. OS/390용 DB2 UDB에서, 단일 SQL문에 대한 실행 구조를 포함하는 플랜 또는 패키지의 세그먼트. 대부분의 SQL문의 경우, 소스 프로그램의 각 SQL문마다 플랜에 하나의 섹션이 있습니다. 그러나, 커서 관련 명령문의 경우, DECLARE, OPEN, FETCH, CLOSE 명령문은 같은 섹션을 참조하므로, 명령문 각각은 DECLARE CURSOR문에서 명명된 SELECT문을 참조합니다. COMMIT, ROLLBACK, 그리고 일부 SET문과 같은 SQL문은 섹션을 사용하지 않습니다.

소스. Data Warehouse Center에서, 단계에 대한 입력이 되는 테이블, 뷰 또는 파일.

소스 서버. DB2 복제에서, 복제 소스 및 Capture 프로그램의 데이터베이스 위치.

소스 유형. 구별 유형을 내부적으로 표시하는 데 사용되는 기존 유형.

소스 테이블. DB2 복제에서, 목표 테이블에 복사될 데이터를 포함하는 테이블. 소스 테이블은 복제 소스 테이블이나 데이터 변경 테이블, 또는 데이터 일관 변경 테이블이 될 수 있습니다. 반의어 - *목표 테이블*.

소스 프로그램. SQL 사전 처리 컴파일러에 의해 처리되는 호스트 언어 명령문과 SQL문의 집합.

소스 함수. 하나 이상의 다른 사용자 정의 함수(UDF)를 구현하기 위해 사용되는 UDF.

소켓. 원격 TCP/IP 상대와 통신하기 위해 TCP/IP 네트워크 응용프로그램에 의해 사용되는 호출 가능한 TCP/IP 프로그래밍 인터페이스.

소프트 점검점. 일부 정보를 로그 파일 헤더에 기록하는프로세스. 이 정보는 데이터베이스 재시작이 필요한 경우 로그의 시작점을 판별하는 데 사용됩니다.

속성. SQL 데이터베이스 설계에서, 엔터티 특성. 예를 들어, 직원 전화번호는 해당되는 직원의 속성 중 하나입니다.

수정 잠금. OS/390용 DB2 UDB에서, MODIFY 속성내에 있는 L-잠금 또는 P-잠금. 이 활동중 잠금 목록은 결합 기능 잠금 구조에 있는 동안 내내 보존됩니다. 요청하는 서브시스템이 실패할 경우, 그 서브시스템의 수정 잠금은 보유된 잠금으로 변환됩니다.

순차 데이터 세트. 하나의 자기 테이프와 같이, 연속되는 물리적 위치를 기준으로 레코드가 구성되는 OS/390용 DB2 UDB 이외의 데이터 세트. 몇 개의 OS/390용 DB2 UDB 데이터베이스 유틸리티에서 순차 데이터 세트를 요구합니다.

순차 프리페치. OS/390용 DB2 UDB에서, 연속 비동기 입출력 조작을 트리거하는 메카니즘. 페이지는 요구되기 전에 페치되고, 몇 개의 페이지가 단일 입출력 조작으로 읽혀집니다.

순환. OS/390용 DB2 UDB에서, 각 테이블이 이전 테이블에 종속되고 첫번째 테이블이 마지막 테이블에 종속되도록 순서화될 수 있는 테이블 세트. 자체 참조 테이블은 단일 구성원에 대한 순환입니다.

순환 공통 테이블 표현식. fullselect로부터 FROM절에 있는 자기 자신을 참조하는 공통 테이블 표현식. 순환 공통 테이블 표현식은 순환 조회를 작성하는 데 사용됩니다.

순환 로그. 사용중인 데이터베이스가 더 이상 필요로 하지 않는 경우 레코드가 겹쳐 쓰일 수 있는 데이터베이스 로그. 결과적으로, 실패가 발생하는 경우, 포워드 복구중에 손실된 데이터를 복원할 수 없습니다. 비교 - *복구가능한 로그*.

순환 조회. 순환 공통 테이블 표현식을 사용하는 fullselect.

순환 주기. 공통 테이블 표현식의 fullselect가 FROM절에 있는 공통 테이블 표현식 이름을 포함할 때 발생하는 순환.

술어. 비교 연산을 표현하거나 의미하는 검색 조건 요소.

스냅샷. *성능 스냅샷* 및 *explain 스냅샷* 참조.

스레드. (1) 일부 운영 시스템에서, 프로세스에서 수행되는 최소 단위 연산. (2) 응용프로그램의 연결을 설명하고, 그 과정을 추적하며, 자원 함수를 처리하고, 액세스 가능성을 OS/390용 DB2 UDB 자원 및 서비스로 구분하는 OS/390용 DB2 UDB 구조. 대부분의 OS/390용 DB2 UDB 함수는 스레드 구조에서 실행됩니다. 비교 - *관련 스레드* 및 *데이터베이스 액세스 스레드*.

스칼라 함수. 다른 값에서 단일 값을 산출하고, 괄호로 묶어 함수 이름 뒤에 인수 목록으로 표현되는 SQL 연산. 반의어 - *컬럼 함수*.

스칼라 fullselect. 단일 값(정확하게 한 컬럼으로 구성되는 데이터의 한 행)을 리턴하는 fullselect.

스케일. 숫자의 분수 부분의 자릿수.

스키마. (1) 테이블, 뷰, 색인 또는 트리거와 같은 데이터베이스 오브젝트 콜렉션. 데이터베이스 스키마는 데이터베이스 오브젝트의 논리적 분류를 제공합니다. (2) OS/390용 DB2 UDB에서, 사용자 정의 함수, 구별 유형, 트리거 및 저장 프로시듀어에 대한 논리적 그룹화. 이러한 유형 중 하나에 해당되는 오브젝트가 작성되면, 그 오브젝트는 하나의 스키마에 지정됩니다. 스키마는 오브젝트의 이름에 의해 판별됩니다. (3) Data Warehouse Center에서, 웨어하우스 목표 테이블과, 웨어하우스 목표 테이블 컬럼들 사이의 관계를 모은 콜렉션으로, 목표 테이블은 하나 이상의 웨어하우스 목표로부터 제공될 수 있습니다.

스타 스키마. 종종 Data Warehouse Center에서 작성되며, OLAP Starter 킷에서 사용되는 관계형 데이터베이스 스키마의 유형.

스택. 임시 레지스터 정보, 매개변수, 부속 루틴의 리턴 주소를 저장하는 메모리 내 영역.

스탭 판. Data Warehouse Center에서, 특정 시간에 웨어하우스 소스에 있는 데이터의 스냅샷.

스토리 보드. 비디오의 비주얼 요약. Video Extender에는 비디오에서 컷을 나타내는 비디오 프레임을 식별하고 저장하기 위해 사용할 수 있는 기능이 포함되어 있습니다. 이러한 표시하는 프레임들은 스토리보드를 빌드하기 위해 사용될 수 있습니다.

시간. 시간, 분, 초를 나타내는 세 부분으로 구성된 값.

시간소인. 연도, 월, 일, 시간, 분, 초, 마이크로초로 표시되는 날짜 및 시간으로 구성되는 일곱개 부분으로 구성된 값.

시간소인 기간. 연도, 월, 일, 시간, 분, 초, 마이크로초를 나타내는 DECIMAL(20,6) 값.

시간종료. 사용할 수 없은 자원으로 인한, OS/390용 DB2 UDB 서브시스템 또는 응용프로그램의 비정상 종료. 설치 스펙은 IRLM 서비스가 시작되기를 OS/390용 DB2 UDB가 기다리는 시간과, 응용프로그램에서 요청하는 자원을 사용할 수 없을 경우 IRLM이 기다리는 시간을 판별하기 위해 설정됩니다. 이 시간 스펙 중 어느 하나를 초과하면, 시간종료가 선언됩니다.

시스템 관리자. 컴퓨터 설치 시 컴퓨터 시스템의 사용을 설계, 제어 및 관리하는 사람.

시스템 네트워크 구조(SNA). 네트워크 구성 및 조작을 제어하기 위한 연속된 조작과 네트워크를 통해 정보 단위를 전송하기 위한 연속된 조작, 프로토콜, 형식, 논리 구조에 대한 설명.

시스템 대화. 분산 처리가 시작되기 전에 시스템 메시지를 처리하기 위해 두 OS/390용 DB2 UDB 서브시스템이 설정해야 하는 대화.

시스템 데이터베이스 디렉토리. 데이터베이스 관리 프로그램을 사용하여 액세스할 수 있는 모든 데이터베이스 항목이 들어 있는 디렉토리. 시스템에서 첫번째 데이터베이스가 작성되거나 카탈로그화될 때 작성됩니다.

시스템 서비스 제어점(SSCP). 종속 노드에 네트워크 서비스를 제공하는 SNA 네트워크내 제어점.

시스템 에이전트. 프리페치 처리, 지연된 쓰기 및 서비스 타스크와 같이, OS/390용 DB2 UDB에서 내부적으로 작성되는 작업 요청.

시스템 진단 작업 영역(SDWA). OS/390 환경에서, 프로그램이나 하드웨어 오류를 설명하는 SYS1.LOGREC 항목에 기록되는 데이터.

시스템 카탈로그. *카탈로그* 참조.

시스템간 결합 기능(XCF). 병렬 Sysplex 내에서 실행하는 권한 부여된 프로그램들 사이의 결합을 지원하기 위해 함수를 제공하는 OS/390의 구성요소.

시스템간 확장 서비스(XES). 병렬 Sysplex 환경에 있는 서로 다른 시스템에서 수행되는 응용프로그램 또는 서브시스템의 여러 인스턴스를 사용 가능하게 하여 결합 기능으로 고성능, 고가용성 데이터 공유를 구현하는 OS/390 서비스 세트.

식별자. OS/390용 DB2 UDB와는 구별되는 주소 공간에 있는 접속 서비스 프로그램이 OS/390용 DB2 UDB에 존재를 알리고 DB2에 연결되는 프로세스를 초기화하기 위해 MVS 서브시스템 인터페이스를 통해 발행하는 요청.

신속 통신 관리 프로그램(FCM). 노드간 통신 지원을 제공하는 함수의 그룹.

실패 구성원 상태. OS/390용 DB2 UDB에서, 데이터 공유 그룹 구성원의 상태. 구성원이 실패하면, XCF는 실패한 구성원 상태를 영구적으로 기록합니다. 이 상태는 보통 구성원의 타스크, 주소 공간 또는 MVS 시스템이 활동중에서 Quiesce로 상태가 변경되기 전에 종료되었음을 의미합니다.

실행 가능 명령문. 응용프로그램에 내포될 수 있고, 동적으로 준비 및 실행되거나 대화식으로 발행될 수 있는 SQL문.

실행 불능 뷰. 다음 상황 중 한 가지가 발생하여 더이상 사용할 수 없는 뷰.

- 뷰가 종속되어 있는 테이블 또는 뷰에 대한 SELECT 특권이 뷰의 정의자로부터 권한 취소되는 경우.
- 뷰 정의가 종속되어 있는 오브젝트가 삭제된 경우(또는 다른 뷰의 경우, 작동 불능으로 된 경우도 가능).

실행 불능 트리거. 삭제되거나 작동 불능 상태가 된 오브젝트나, 권한 취소된 특권에 의존하는 트리거.

실행 불능 패키지. 종속되어 있는 함수가 삭제되어서 사용할 수 없는 패키지. 그러한 패키지는 명시적으로 리바인드되어야 합니다. 반의어 - *유효하지 않은 패키지*.

실행 취소. 복구할 수 있는 OS/390용 DB2 UDB 자원에 대해 복구 단위별로 수행된 변경사항이 백아웃되어야 함을 나타내는 복구 단위 상태.

아

아카이브 로그. (1) 일반 프로세스에 더 이상 필요없어 닫힌 로그 파일 세트. 이 파일들은 롤 포워드 복구에서 사용하기 위해 보유되어 있습니다. 반의어 - *사용중인 로그*. (2) 사용중인 로그로부터 복사되는 로그 레코드를 포함하는 OS/390용 DB2 UDB 로그의 부분.

알려진 주소. 노드간 연결을 설정하기 위해 네트워크에 있는 특정 노드를 고유하게 식별하는 데 사용하는 주소. 잘 알려진 주소는 논리 노드에서 사용되는 포트와 네트워크 주소를 결합한 것입니다.

압축. 테이블에 데이터에 대한 변경사항 실행기록이 아닌 현재 데이터가 들어 있음을 나타내는 테이블 속성. 압축 테이블에는 테이블에 있는 각 기본 키 값에 대해 한 행 이상이 포함되어 있습니다. 결과적으로, 압축된 테이블을 사용하여 화면 갱신을 위해 현재 정보를 제공할 수 있습니다.

압축 사전. OS/390용 DB2 UDB에서, 압축 및 압축해제 프로세스를 제어하는 사전. 이 사전은 테이블 공간이나 테이블 공간 파티션의 데이터로부터 작성됩니다.

압축되지 않은 CCD 테이블. DB2 복제에서, 한 행에 대한 값이 변경되는 실행기록을 포함하는 CCD 테이블. 이러한 유형의 테이블은 감사할 때 유용합니다. 반의어 - **압축된 CCD 테이블**.

압축된 CCD 테이블. DB2 복제에서, 한 행의 최근 값만 포함하는 CCD 테이블. 이러한 유형의 테이블은 원격 위치에 변경사항을 스테이징하고 핫스폿 갱신사항을 요약할 때 유용합니다. 반의어 - **압축되지 않은 CCD 테이블**.

액세스 경로. (1) 특정 테이블로부터 데이터를 검색하기 위해 최적화 알고리즘에 의해 선택되는 방법. 예를 들어, 액세스 경로는 색인, 순차적 스캔 또는 이 둘을 조합하여 사용할 수 있습니다. (2) SQL문에 지정된 데이터를 찾기 위해 사용되는 경로. 액세스 경로는 색인화되거나 순차적으로 될 수 있습니다.

액세스 메소드 서비스. VSAM 키순 데이터 세트를 정의하고 다시 생성하기 위해 사용되는 기능.

액세스 플랜. 특정 SQL문을 평가하기 위해 최적화 알고리즘에 의해 선택되는 액세스 경로 세트. 액세스 플랜은 조작 순서를 지정하여 실행 계획, 실현 방법(예, JOIN), 명령문에서 참조되는 각 테이블에 대한 액세스 경로 등을 분석합니다.

액세스 함수. 컬럼에 저장된 텍스트의 데이터 유형을 Text Extender에서 처리될 수 있는 유형으로 변환하는 사용자가 제공하는 함수.

언어 환경®. Net.Data 매크로에서 DB2와 같은 외부 데이터 소스로, 또는 Perl과 같은 프로그래밍 언어로의 액세스를 제공하는 모듈.

에이전트. (1) 특정 클라이언트 응용프로그램에 의한 모든 DB2 요청을 수행하는 별도의 프로세스 또는 스레드. (2) OS/390용 DB2 UDB에서, OS/390용 DB2 UDB 작업 단위(UOW)에 관련된 모든 프로세스를 연관시키는 구조. *연합 에이전트*는 보통 *연합 스레드*의 동의어입니다. *시스템 에이전트*는 프리페치 처리, 지연된 쓰기 및 서비스 타스크와 같은 연합 에이전트와는 독립적으로 처리하는 작업 단위(UOW)입니다.

에이전트 사이트. Data Warehouse Center에서, 단일 네트워크 호스트 이름으로 정의된 위치로, 에이전트 응용프로그램이 설치되어 있습니다.

여유 공간. OS/390용 DB2 UDB에서, 페이지에서 사용되지 않은 총 공간량. 레코드나 제어 정보를 저장하는데 사용되지 않는 공간이 여유 공간입니다.

연결. (1) 응용프로그램 프로세스와 응용프로그램 서버(AS)와의 연결. (2) 데이터 통신에 있어서, 정보를 운반하기 위해 기능 장치간에 연결이 설정됩니다. (3) SNA에서, 정보가 교환될 수 있는 두 상대 LU 사이의 통신 경로 존재(예를 들어, 대화 방법에 의해 연결되고 통신하는 두 개의 OS/390용 DB2 UDB 서브시스템).

연결. DB2에서, 데이터베이스 레벨에서의 오브젝트 액세스.

연결 관리 프로그램. Live Connection을 지원하기 위해 필요한 Net.Data의 실행 파일인 dtwcm.

연결 핸들. CLO 내에서, 연결과 관련된 정보가 들어 있는 데이터 오브젝트. 이 정보에는 일반 상태 정보, 트랜잭션 상태 및 진단 정보가 포함됩니다.

연결 ID. OS/390용 DB2 UDB에서, 접속 기능에 의해 제공되고 특정 주소 공간 연결과 연관되는 식별자.

연기된 embedded SQL. OS/390용 DB2 UDB에서, 완전히 정적이지도 않고 완전히 동적이지도 않은 SQL문. 정적 명령문처럼 응용프로그램 내에 삽입되지만, 동적 명령문과 같이 응용프로그램이 실행되는 동안 준비됩니다.

연속 삭제. SQL에서, 테이블 P에 종속되거나, 테이블 P 연쇄로부터 조작을 삭제하는 테이블에 종속되는 테이블.

연쇄. Data Warehouse Center에서, 일련의 이벤트를 실행하는 것. 한 단계에서 다른 단계로 연속될 때, 그 단계들은 순차적으로나 동시에 실행됩니다. 또한, 단계는 단계의 실행이 종료된 후에 실행되는 프로그램으로 연속될 수도 있습니다.

연쇄 거부. DB2 복제에서, 충돌이 검출되어 그 자체가 거부된 트랜잭션과 연관되어 있어서 복제 트랜잭션을 거부하는 프로세스.

연쇄 삭제. OS/390용 DB2 UDB가 삭제된 상위 행의 모든 하위 행을 삭제할 때 참조 제한조건을 시행하는 방법.

연합 데이터베이스 시스템. (1) DB2 서버와 그 서버가 조회를 보내는 여러 데이터 소스. 연합 데이터베이스 시스템에서, 클라이언트 응용프로그램은 하나의 SQL문을 사용하여 여러 데이터베이스 관리 시스템에서 분산되어 있는 데이터를 조인하고 그 데이터를 지역 데이터인 것처럼 볼 수 있습니다. (2) 다음으로 구성된 분산 컴퓨팅 시스템.

- **연합 서버**라고 하는 DB2 서버.
- 연합 서버가 조회를 보내는 여러 개의 데이터 소스.

  각 데이터 소스는 관계형 데이터베이스 관리 시스템의 인스턴스에 더하여 인스턴스가 지원하는 데이터베이스들로 구성됩니다.

  데이터 소스는 반자동일 수 있습니다. 예를 들어, 연합 서버는 Oracle 응용프로그램이 이들 데이터 소스에 액세스할 수 있는 것과 동시에 Oracle 데이터 소스로 조회를 보낼 수 있습니다.

예약어. (1) 프로그램이나 컴파일러가 취하는 조치를 서술하기 위해 소스 프로그램이 사용하는 단어. 사용자 정의 이름이나 시스템 이름으로서 프로그램에 나타날 수 없습니다. (2) SQL 표준에서 특수 사용을 위해 예약된 단어.

예외 테이블. OS/390용 DB2 UDB에서, CHECK DATA 유틸리티가 찾는 참조 제한조건이나 테이블 점검 제한조건을 위반하는 행을 보유하는 테이블.

오류 페이지 범위. 실제로 손상된 것으로 간주되는 페이지 범위. OS/390용 DB2 UDB는 어떤 사용자도 이 범위에 속하는 페이지에 액세스할 수 없게 합니다.

오른쪽 외부 조인. OS/390용 DB2 UDB에서, 조인되는 두 테이블의 일치되는 행을 포함하고 두 번째 조인 피연산자의 일치되지 않는 행은 보존하는 조인 조작의 결과. *조인* 참조.

오버로드된 함수 이름. 함수 경로나 스키마 내에 여러 함수가 존재하는 함수 이름. 동일한 스키마 내에 있는 함수 이름들은 서로 다른 시그니처를 가집니다.

오버플로우 레코드. (1) 간접 주소지정 파일에서, 그 키가 전체 트랙 주소 또는 홈 레코드 주소로 임의지정(randomized)되는 레코드. (2) DB2에서, 현재 저장된 페이지에 들어갈 만큼 큰 갱신된 레코드. 이 레코드는 다른 페이지로 복사되고,

원래 위치는 새 위치를 가리키는 포인터로 대체됩니다. (3) 데이터베이스 모니터에서, Named Pipe가 가득차서 레코드가 제 시간에 처리되지 않았기 때문에 레코드가 버렸음을 나타내기 위해 이벤트 모니터 데이터 스트림에 삽입되는 레코드. 오버플로우 레코드는 버려진 레코드 수를 표시합니다.

오브젝트. (1) SQL로 조작 또는 작성될 수 있는 것(예, 테이블, 뷰, 색인, 패키지 등). (2) 오브젝트 지향 설계 또는 프로그래밍에서, 데이터 및 그 데이터와 연관된 조작으로 구성되는 도출. (3) NetWare의 경우, 네트워크에서 정의되어 파일 서버에 대한 액세스가 제공되는 엔터티.

오브젝트 등록 정보. 오브젝트와 연관되는 정보의 범주를 식별하는 등록 정보. NetWare 바인더리 오브젝트에는 하나 이상의 등록 정보가 지정될 수 있습니다. DB2 서버 인스턴스 오브젝트에는 오브젝트 내에서 레코드의 위치를 표시하는 오브젝트 등록 정보인 NET_ADDR이 있습니다.

오브젝트 유형. (1) NetWare 파일 서버의 바인더리에서 오브젝트를 분류하는 2 바이트 숫자. 062B는 DB2 데이터베이스 서버 오브젝트 유형을 나타냅니다. (2) 유사한 조치 및 특성을 공유하는 오브젝트 인스턴스들의 범주 또는 그룹.

오프라인 백업. 테이블 공간 또는 데이터베이스가 응용프로그램에 의해 액세스되지 않을 때 이루어진 테이블 공간 또는 데이터베이스의 백업. 백업 데이터베이스 유틸리티는 백업이 완료될 때까지 데이터베이스에 대한 독점 사용을 합니다. 반의어 - *온라인 백업*.

오프라인 복원. 백업으로부터 테이블 공간이나 데이터베이스 사본을 복원하는 것. 백업 데이터베이스 유틸리티는 복원이 완료될 때까지 데이터베이스에 대한 독점 사용을 합니다. 반의어 - *온라인 복원*.

온라인 모니터. *성능 모니터*에서 참조하십시오.

온라인 백업. 데이터베이스나 테이블 공간이 다른 응용프로그램에 의해 액세스되는 동안 이루어지는 테이블 공간 또는 데이터베이스의 백업. 반의어 - *오프라인 백업*.

온라인 복원. 데이터베이스나 테이블 공간이 다른 응용프로그램에 의해 액세스되는 동안 이루어지는 테이블 공간 또는 데이터베이스 사본의 복원. 반의어 - *오프라인 복원*.

온라인 분석 처리(OLAP). OLAP Starter 킷에서, 조정된 엔터프라이즈 데이터를 실시간으로 분석해야 하는 사용자를 위한 다차원, 다중 사용자, 클라이언트 서버 컴퓨팅 환경.

온라인 자원 정의. CICS가 있는 OS/390 환경에서, 테이블을 어셈블링하지 않고 CICS 자원을 온라인으로 정의하기 위해 사용하는 기능.

올림 테이블. DB2 복제에서, 여러 목표 테이블에 대해 데이터를 갱신하기 위한 소스로 사용될 수 있는 CCD 테이블.

완료 요구. 데이터 무결성을 유지하기 위해 전체 조작이 완료되어야 하는 OS/390용 DB2 UDB 처리 동안의 상태.

완전하지 않은 CCD 테이블. DB2 복제에서, 작성될 때 비어 있고 소스가 변경될 때마다 행이 추가되는 CCD 테이블. 반의어 - **완전한 CCD 테이블**.

완전한 CCD 테이블. 소스 테이블이나 뷰로부터의 술어와 소스 뷰를 만족시키는 모든 행을 포함하는 CCD 테이블. 반의어 - **불완전한 CCD 테이블**.

완전한 LU 이름. *네트워크 규정 이름* 참조.

외부 갱신. 목표 테이블에 적용되어 지역 테이블로 복사된 갱신.

외부 루틴. OS/390용 DB2 UDB에서, 외부 프로그래밍 언어로 작성되는 코드를 기초로 하는 사용자 정의 함수(UDF) 또는 저장 프로시듀어.

외부 조인. (1) 조인되는 모든 테이블들에 공통되지 않은 컬럼이 결과 테이블의 일부가 되는 조인 방법. 반의어 - *내부 조인*. (2) OS/390용 DB2 UDB에서, 조인되는 두 테이블의 일치되는 행을 포함하고 조인되는 테이블의 일치되지 않는 행의 일부나 모두는 보존하는 조인 조작의 결과. 또한 *조인* 참조.

외부 함수. OS/390용 DB2 UDB에서, 본문이 스칼라 인수 값을 취하여 각 호출마다 스칼라 결과를 생성하는 프로그래밍 언어로 작성되는 함수. 반의어 - *전래 함수* 및 *내장 함수*.

외부 CCD 테이블. DB2 복제에서, 등록된 복제 소스여서 직접 복사 작업이 가능한 CCD 테이블. 이 테이블은 SOURCE_OWNER 및 SOURCE_TABLE로도 언급되는 등록 테이블에 고유한 행을 가지고 있습니다. 반의어 - **내부 CCD 테이블**.

왼쪽 외부 조인. OS/390용 DB2 UDB에서, 조인되는 두 테이블의 일치되는 행을 포함하고 두 테이블의 일치되지 않는 행은 보존하는 조인 조작의 결과.*조인* 및 *오른쪽 외부 조인* 참조.

요구 시 타이밍. 간혹 연결되는 시스템에 대한 복제 타이밍을 제어하는 방법. Capture 및 Apply 프로그램을 조작하기 위해 ASNSAT 프로그램을 사용해야 합니다. 반의어 - **이벤트 타이밍** 및 **간격 타이밍**.

요청 확약. OS/390용 DB2 UDB에서, 구성원이 데이터를 수정하여 확약 또는 구간 복원할 준비가 된 경우에 준비 단계에 제출되는 투표.

원격. OS/390용 DB2 UDB에서, 원격 DB2 서브시스템에 의해 유지보수되는 오브젝트. 예를 들어, 원격 뷰는 원격 DB2 서브시스템에 의해 유지보수되는 뷰입니다. 반의어 - *지역*.

원격 데이터베이스. 사용중인 데이터베이스가 아닌, 워크스테이션에 물리적으로 위치한 데이터베이스. 반의어 - *지역 데이터베이스*.

원격 서브시스템. OS/390용 DB2 UDB에서, *지역 서브시스템*을 제외하고, 사용자나 응용프로그램이 통신할 수 있는 RDBMS. 서브시스템은 물리적 측면에서 원격일 필요가 없으며, 같은 OS/390 시스템에 있는 같은 프로세서에서도 작동될 수 있습니다.

원격 작업 단위(RUOW). SQL문의 원격 준비 및 실행을 허용하는 작업 단위(UOW).

원격 접속 요청. OS/390용 DB2 UDB에서, 지역 DB2 서브시스템에 접속하기 위해 원격 위치에서 이루어진 요청. 특히, 송신되는 요청은 SNA 함수 관리 머리글 5입니다.

원래 타스크. OS/390용 DB2 UDB에서, 조회의 부분들을 병렬로 실행하는 다른 실행 단위(*병렬 타스크*라고 함)으로부터 데이터를 수신하는 병렬 그룹의 1차 에이전트.

웜 스타트. (1) 이전에 초기화된 입출력 작업 대기행렬의 재사용을 허용하는 재시작. 반의어 - *콜드 시동*. (2) DB2 복제에서, 이전에 초기화된 입력 및 출력 작업 대기행렬을 재사용할 수 있게 하는 Capture 프로그램의 시작.

웨어하우스. 전략 결정을 지원하기 위해 사용되는 주제 중심의 비휘발성 데이터 콜렉션. 웨어하우스는 비지니스 인텔리전스에 대한 데이터 통합의 중심 지점입니다. 이것은 엔터프라이즈 내에서 데이터마트에 대한 데이터 소스로, 엔터프라이즈 데이터의 공통 뷰를 전달합니다.

웨어하우스 목표. Data Warehouse Center에서 관리되는 단일 데이터베이스의 테이블, 색인 및 별명의 부속 집합.

웨어하우스 소스. Data Warehouse Center에 대해 정의된 단일 데이터베이스의 테이블 및 뷰 부속 집합이나, 파일 세트.

웨어하우스 에이전트. Data Warehouse Center에서, 데이터 이동과 변환을 관리하는 런타임 프로세스.

웨어하우스 제어 데이터베이스. Data Warehouse Center 메타데이터에 필요한 제어 테이블을 포함하는 Data Warehouse Center 데이터베이스.

웨어하우스 프로그램 그룹. Data Warehouse Center에서, 프로그램 오브젝트를 보유하는 컨테이너(폴더).

위성. 위성 제어 데이터베이스에서 해당 그룹과 동기화되는 DB2 서버가 있는, 간혹 연결되는 클라이언트.

위성 관리 센터. 위성에 대한 중앙 집중화된 관리 지원을 제공하는 사용자 인터페이스.

위성 제어 서버. 위성 제어 데이터베이스인 SATCTLDB를 포함하는 DB2 Universal Database.

위치 경로. XPath에 의해 정의된 위치 경로에 대한 약어화된 구문의 부속 집합. XML 요소나 속성을 식별하기 위한 일련의 XML 태그. 이것은 추출될 주제를 식별하기 위해 사용자 정의 함수를 추출할 때 사용되며, 검색 기준을 식별하기 위해 Text Extender의 검색 사용자 정의 함수에서 사용됩니다.

위치 이름. OS/390용 DB2 UDB이 서브시스템 네트워크에서 특정 DB2 서브시스템을 언급하는 이름. 반의어 - *LU 이름*.

위치 지정자. *LOB 위치 지정자* 참조.

유니코드. ISO 10646 표준 부속 집합인 국제 문자 코드화 체계. 지원되는 각 문자는 고유한 2바이트 코드를 사용하여 정의됩니다.

유형 1 색인. MVS/ESA용 DB2 버전 4 릴리스에서 작성되었거나, 버전 4에서 유형 1 색인으로 지정되는 색인. 반의어 - *유형 2 색인*. OS/390용 DB2 UDB 버전 7 현재, 유형 1 색인은 더이상 지원되지 않습니다.

유형 2 색인. OS/390용 DB2 버전 6 이후에 DB2 릴리스에서 작성되거나 버전 4 또는 버전 6에서 유형 2 색인으로 지정된 색인. 반의어 - *유형 1 색인*.

유형변환 함수. 데이터 유형(origin) 인스턴스를 다른 데이터 유형(목표)으로 변환하는 데 사용되는 함수. 일반적으로 유형변환 함수에는 목표 데이터 유형 이름이 있습니다. 그 함수에는 유형이 원래의 데이터 유형인 단일 인수가 있으며, 리턴 유형은 목표 데이터 유형입니다.

유효하지 않은 패키지. 패키지가 종속된 오브젝트가 삭제될 때 무효가 되는 패키지. (그 오브젝트의 유형은 함수 이외의 다른 유형(예: 색인)입니다.) 그러한 패키지는 암시적으로 호출시 리바인드됩니다. 반의어 - *작동 불능 패키지*.

윤곽. OLAP Starter 킷에서, OLAP Starter 킷 내에서 모든 데이터베이스 요소를 정의하는 구조. 예를 들어, 윤곽에는 차원, 구성원 및 공식의 정의가 포함됩니다.

음영처리. 현재 페이지 내용을 겹쳐 쓸 수 없는 복구 기술. 대신, 트랜잭션 구간 복원으로 인해 시스템 상태 복원을 지원하는 데 더 이상 필요없을 때까지, 값이 교체 중인 페이지가 음영 사본으로 보유되어 있는 동안 새 페이지가 할당 및 작성됩니다.

응용프로그램. 타스크를 수행하는 하나의 프로그램이나 일련의 프로그램 세트(예: 급여 지불 응용프로그램).

응용프로그램 리퀘스터. 응용프로그램 프로세스로부터 데이터베이스 요청을 얻고 응용프로그램 서버(AS)로 전달하는 기능.

응용프로그램 서버(AS). 응용프로그램 프로세스가 연결되어 있는 지역 또는 원격 데이터베이스 관리 프로그램.

응용프로그램 프로세스. 자원 및 잠금이 할당되는 단위. 응용프로그램 프로세스에는 하나 이상의 프로그램 수행이 포함됩니다.

응용프로그램 플랜. 바인드 프로세스 동안 생성되는 제어 구조. OS/390용 DB2 UDB는 명령문 실행 동안 발견하는 SQL문을 처리하기 위해 응용프로그램 플랜을 사용합니다.

응용프로그램 ID. 네트워크를 통해 응용프로그램을 고유하게 식별하는 문자열. 응용프로그램이 데이터베이스에 연결할 때 ID가 생성됩니다. 이 ID는 클라이언트와 서버 모두에 알려지며, 이 ID를 사용하여 응용프로그램의 두 부분을 상관시킬 수 있습니다.

이동 복제 모드. 자발적으로나 지속적으로가 아닌, 필요에 따라 Capture 및 Apply 프로그램이 작동하는 복제 모드. 이 모드든 모빌 클라이언트로부터 호출되고, 이 모드에서는 모빌 클라이언트를 소스나 목표 서버로의 연결에 사용할 수 있을 때 데이터를 복제할 수 있습니다.

이동 복제 인에이블러. 이동 클라이언트에서 이동 복제 모드를 시작하는 복제 프로그램.

이동 클라이언트. 모빌 환경에 사용되는 모빌 인에이블러, 복제 소스 및 목표 테이블이 있는 노드(보통, 랩톱 컴퓨터). 이동 복제 모드는 이동 클라이언트로부터 호출됩니다.

이미지 내용별 조회(QBIC). 보통 색상이나 바탕과 같이, 사용자가 비주얼 특성에 의해 이미지를 검색할 수 있도록 Image Extender에서 제공되는 기능.

이미지 복사. 테이블 공간 전체나 부분에 대한 완전한 재생성. OS/390용 DB2 UDB는 전체 테이블 공산을 복사하기 위해 전체 이미지 사본을 만들거나, 마지막 이미지 복사 이후로 수정된 페이지만 복사하기 위해 증분식 이미지 사본을 만들기 위한 유틸리티 프로그램을 제공합니다.

이벤트 모니터. 일정 기간에 걸친 데이터베이스 활동시 데이터의 모니터 및 수집을 위한 데이터베이스 오브젝트.

이벤트 타이밍. DB2 복제에서, 복사 작업 내역 순환을 시작하는 시기를 가장 정밀하게 제어하는 방법. 이벤트를 처리할 시간과 이벤트를 지정해야 합니다. 반의어 - **간격 타이밍** 및 **요구 시 타이밍**.

이상 종료. *타스크의 비정상 종료*를 참조하십시오.

이상 종료 이유 코드. OS/390용 DB2 UDB에서의 문제점을 고유하게 식별하는 4 바이트의 16진수 코드.

이전 트리거. OS/390용 DB2 UDB에서, 트리거 활성화 시간 BEFORE로 정의된 트리거.

이주. (1) 데이터를 변환하지 않고 다른 컴퓨터 시스템으로 데이터를 이동시키는 프로세스. (2) 이전 버전 또는 릴리스를 대체하기 위해 프로그램의 새 버전 또는 릴리스를 설치하는 것. (3) 기존의 OS/390용 DB2 UDB 서브시스템을 갱신된 릴리스나 현재 릴리스로 변환하는 프로세스. 이 프로세스에서, 이전 릴리스에서 작성한 데이터를 유실하지 않고 갱신된 릴리스나 현재 릴리스의 함수를 확보할 수 있습니다.

인수. 런타임 함수 또는 프로시듀어로부터 리턴되거나 전달되는 값.

인스턴스. (1) *데이터베이스 관리 프로그램 인스턴스* 참조. (2) 논리적 DB2 extender 서버 환경. 같은 워크스테이션에 몇 개의 DB2 Extender 서버 인스턴스가 있을 수 있지만, 각 DB2 인스턴스마다 단 하나의 인스턴스가 있습니다.

인용된 이름. *분리 식별자* 참조.

인접 노드. 다른 어떤 노드와도 연결되지 않은 최소한 한 경로에 의해 연결되는 두 노드.

인증됨. 변환 할당 시 OS/390용 DB2 UDB에서 사용자의 확인된 권한 부여 ID를 제공할 수 있게 하는 LU 6.2 보안 옵션. 상대 서브시스템에서는 사용자의 유효성을 확인할 수 없습니다.

인터넷 프로토콜(IP). 인터넷 환경에서 소스로부터 목적지로 데이터의 경로를 지정하는 데 사용되는 프로토콜.

일관성 지점. 프로그램이 액세스하는 복구 가능한 모든 데이터가 일관된 지점. 갱신, 삽입 및 삭제가 물리적 데이터베이스로 확약되거나 구간 복원될 때 일관성 지점이 발생합니다. 동의어 - *확약 지점* 및 *동기 지점*.

일관성 토큰. OS/390용 DB2 UDB에서, 응용프로그램에 대한 버전 식별자를 생성하기 위해 사용되는 시간소인.

일반 자원 이름. OS/390 환경에서, 병렬 Sysplex 환경에서 세션 분산 및 균형화를 처리하기 위해 같은 함수를 제공하는 몇 개의 응용프로그램을 나타내기 위해 VTAM에서 사용하는 이름.

일반 테이블 공간. 비임시 데이터를 저장할 수 있는 테이블공간.

읽기 안정성(RS). 응용프로그램이 트랜잭션 내에서 검색하는 행만 잠그는 분리 레벨. 읽기 안정성은 트랜잭션 동안 읽혀지는 규정 행이, 트랜잭션이 완료될 때까지 다른 응용프로그램 프로세스에 의해 변경되지 않도록 하고, 다른 응용프로그램 프로세스에 의해 변경되는 행은 변경이 해당 프로세스에 의해 확약될 때까지 읽혀지지 않도록 합니다. 읽기 안정성(RS)은 반복 읽기(RR)보다는 많고 커서 안정성(CS)보다는 적은 동시성을 허용합니다.

임시 테이블. 중간 결과를 보유하기 위해 SQL문을 처리하는 동안 작성되는 테이블. 반의어 - *결과 테이블*.

임시 테이블 공간. 임시 테이블만 저장할 수 있는 테이블 공간.

입력된 매개변수 표시문자. 목표 데이터 유형과 함께 지정된 매개변수 표시문자. 다음과 같은 일반 형태를 지닙니다.

```
CAST(? AS data-type)
```

자

자동 리바인드. (1) **bind** 명령을 수동으로 입력하거나 바인드 파일이 존재해야 할 필요없이, 자동으로 무효화된 패키지를 리바인드하는 기능. (2) OS/390용 DB2 UDB에서, 응용프로그램 프로세스가 실행을 시작하고 필요로 하는 바인드된 응용프로그램 플랜 또는 패키지가 유효하지 않을 경우에 사용자가 BIND 명령을 발행하지 않고도 자동으로 SQL문이 바인드되는 프로세스. 또한 *BIND* 참조.

자동 확약. 각 SQL문 뒤에 현재의 작업 단위(UOW)를 자동으로 확약하는 것.

자원. OS/390용 DB2 UDB에서, 잠그거나 청구하는 오브젝트로, 테이블 공간, 색인 공간, 데이터 파티션, 색인 파티션 또는 논리적 파티션이 될 수 있습니다.

자원 제어 테이블(RCT). CICS가 있는 OS/390용 DB2 UDB에서, 트랜잭션이나 트랜잭션 그룹에 대한 권한 부여 및 액세스 속성을 정의하는, 사이트 제공 매크로 매개변수에 의해 작성되는 CICS 접속 기능 구성.

자원 한계 기능(RLF). 동적 조작 SQL문이 지정된 시간 한계를 초과하지 않도록 하는 OS/390용 DB2 UDB 코드의 부분. 동의어 - *조정자*.

자원 한계 스펙 테이블. OS/390용 DB2 UDB에서, 자원 한계 기능에 의해 시행될 한계를 지정하는 사이트 정의 테이블.

자원 할당. OS/390용 DB2 UDB에서, 특별히 데이터베이스 자원으로 취급되는 플랜 할당의 부분.

자체 참조 부속 조회. SQL문의 오브젝트인 동일한 테이블을 참조하는 DELETE, INSERT 또는 UPDATE문 내부의 전체선택(fullselect) 또는 부속 선택.

자체 참조 제한조건. OS/390용 DB2 UDB에서, 테이블이 자체에 종속되는 관계를 정의하는 참조 제한조건.

자체 참조 테이블. 동일한 참조 제한조건에 있는 상위 및 종속 테이블인 테이블.

자체 참조 행. 자신의 상위인 행.

작업 단위(UOW). 응용프로그램 프로세스 내에서의 복구가능한 일련의 조작. 언제든지 응용프로그램 프로세스는 하나의 작업 단위이지만, 확약 또는 구간 복원 조작 결과, 응용프로그램 프로세스의 활동 기간에는 여러 개의 작업 단위가 포함될 수 있습니다. OS/390용 DB2 UDB *다중 사이트 갱신* 조작에서, 단일 작업 단위(UOW)에는 몇 개의 *복구 단위*가 포함될 수 있습니다. 동의어 - *트랜잭션*.

작업 스케줄러. 데이터베이스 작업 수행 및 관리를 위해 특정 타스크를 자동화하는 데 사용되는 프로그램.

작업 제어 언어(JCL). 운영 체제에 대해 작업을 식별하고 작업의 요구사항을 설명하기 위해 사용되는 제어 언어.

작업 파일. DB2 복제에서, 복사 작업 내역 세트를 처리할 때 Apply 프로그램에서 사용되는 임시 파일.

작업 항목 서브시스템(JES). 시스템 내로 작업을 수신하고 작업에 의해 생성되는 모든 출력을 처리하는 IBM 사용권 부여 프로그램.

잠금. 데이터베이스 관리 프로그램이 데이터의 무결성을 확인하기 위해 사용하는 기제. 잠금은 동시에 여러 사용자가 일관성 없는 데이터에 액세스하지 못하도록 합니다.

잠금. (1) 데이터에 액세스하거나 이벤트를 순서화하는 방법. (2) 어떤 응용프로그램 프로세스가 다른 응용프로그램 프로세스의 미확약 변경사항을 인식하지 못하도록 하고, 어떤 프로세스가 액세스하는 데이터를 다른 응용프로그램 프로세스가 갱신하지 못하도록 하는 방법. (3) 데이터에 대한 액세스나 동시 이벤트를 제어하는 수단. OS/390용 DB2 UDB 잠금은 IRLM에서 수행됩니다.

잠금 구조. OS/390용 DB2 UDB에서, 논리 자원에 대한 공유 잠금과 독점 잠금을 지원하기 위한 일련의 잠금 항목으로 구성되는 결합 기능 데이터 구조.

잠금 레벨 자동 업그레이드. 데이터베이스 관리 프로그램에서, 한 에이전트에 대해 발행되는 잠금 수가 데이터베이스 구성에 지정된 한계를 초과할 때 발생하는 응답. 이 제한은 MAXLOCKS 구성 매개변수에 의해 정의됩니다. 잠금 레벨 자동 업그레이드 중에 테이블 행에 있는 잠금을 테이블의 잠금으로 변환함으로써 잠금이 해제됩니다. 이는 한계가 더 이상 초과되지 않을 때까지 반복됩니다.

잠금 모드. OS/390용 DB2 UDB 잠금이 보유되어 있는 자원에 대해 동시에 수행 중인 프로그램들이 가질 수 있는 액세스 유형에 대한 표시.

잠금 승격. OS/390용 DB2 UDB 잠금 크기나 모드를 상위 레벨로 변경하는 프로세스.

잠금 오브젝트. OS/390용 DB2 UDB 잠금에 의해 제어되는 자원.

잠금 지속기간. OS/390용 DB2 UDB 잠금이 보유되는 간격.

잠금 크기. 테이블 데이터에서 OS/390용 DB2 UDB 잠금에 의해 제어되는 데이터의 양으로, 행, 페이지, LOB, 파티션, 테이블 또는 테이블 공간이 될 수 있습니다.

잠금해제. 이전에 잠긴 오브젝트나 시스템 자원을 해제하고 이를 OS/390용 DB2 UDB 내에서 일반적으로 사용할 수 있는 상태로 되돌리는 조작.

장치 이름. 특정 장치를 참조하는 장치 드라이버 또는 시스템에 의해 예약된 이름.

재시작 보류(RESTP). OS/390용 DB2 UDB에서, 오브젝트에 대해 재시작(백아웃) 작업을 수행해야 함을 나타내는 페이지 세트 또는 파티션의 제한된 상태. 페이지 세트나 파티션에 대한 모든 액세스는 RECOVER POSTPONED 명령이나, 시스템 매개변수가 LBACKOUT=AUTO일 경우 재시작 후에 OS/390용 DB2 UDB에서 호출하는 자동 온라인 백아웃에 의해 액세스되는 것을 제외하고 거부됩니다.

재이주. 이전 릴리스로 후진한 후에 OS/390용 DB2 UDB의 현재 릴리스로 리턴하는 프로세스. 이 프로시듀어는 다른 이주 프로세스로 구성됩니다.

재최적화. 런타임 시 SQL문의 경로의 액세스 경로를 다시 고려하는 OS/390용 DB2 UDB 프로세스. 재최적화 동안, OS/390용 DB2 UDB는 호스트 변수, 매개변수 표시문자 또는 특수 레지스터의 값을 사용합니다.

저장 프로시듀어. (1) 데이터베이스에 저장되어 이름으로 호출할 수 있는 Embedded SQL문 및 절차 구성 블록. 저장 프로시듀어는 응용프로그램이 두 부분에서 수행될 수 있도록 합니다. 한 부분은 클라이언트에서 수행되고 다른 부분은 서버에서 수행됩니다. 이로써 하나의 호출로 데이터베이스를 여러 번 액세스할 수 있습니다. 동의어 - *프로시듀어*. (2) OS/390용 DB2 UDB에서, SQL CALL문 사용을 통해 시작될 수 있는 사용자 작성 응용프로그램.

저장영역 그룹. OS/390용 DB2 UDB 데이터가 저장될 수 있는 명명된 DASD 볼륨 세트.

전래 함수. OS/390용 DB2 UDB에서, 이미 데이터베이스 관리 프로그램에 알려진 또 다른 내장 함수나 사용자 정의 함수에 의해 구현되는 함수. 이 함수는 스칼라 함수나 컬럼(총계) 함수가 될 수 있으며, 값 집합으로부터 단일 값을 리턴합니다(예, MAX, AVG). 반의어 - *외부 함수* 및 *내장 함수*.

전방향 로그 복구. OS/390용 DB2 UDB가 모든 REDO 로그 레코드를 적용하기 위해 전방향으로 로그를 처리하는 재시작 처리의 세 번째 단계.

전송 제어 프로토콜/인터넷 프로토콜(TCP/IP). 근거리 통신망 및 광역망 모두에 피어 투 피어 연결 함수를 제공하는 통신 프로토콜 집합.

전역 잠금. OS/390용 DB2 UDB에서, DB2 서브시스템 내에서와 DB2 서브시스템 사이에 동시 제어를 제공하는 잠금. 잠금 범위는 데이터 공유 그룹의 모든 DB2 서브시스템입니다.

전역 잠금 경합. 데이터 공유 그룹의 서로 다른 OS/390용 DB2 UDB 구성원들이 공유 자원을 순차화하려고 할 때의 그 구성원 사이의 잠금 요청 충돌.

전이 테이블. 트리거 수정에 의해 영향받는 각 행에 대한 전이 값이 들어 있는 명명된 임시 테이블. 기존의 전이 테이블에는 수정이 적용되기 전에 영향받은 행의 값들이 포함되고, 새로운 전이 테이블에는 수정이 적용된 후 영향받은 행의 값들이 포함됩니다.

전체 새로 고침. DB2 복제에서, 기존 데이터를 대체하면서, 사용자 테이블에서 관계가 있는 모든 데이터가 목표 테이블로 복사되는 프로세스. 반의어 - *부분 새로 고침*.

전체 외부 조인. 조인되는 두 테이블의 일치되는 행을 포함하고 두 테이블의 일치되지 않는 행은 보존하는 SQL 조인 조작의 결과. *조인* 참조.

전환 변수. FOR EACH ROW 트리거에서만 유효한 변수. 현재 행에 대한 전이 값으로의 액세스를 허용합니다. 기존의 전이 변수는 수정이 적용되기 전의 행 값이고, 새로운 전이 변수는 수정이 적용된 후의 행 값입니다.

절. OS/390용 DB2 UDB SQL에서, SELECT 절이나 WHERE 절과 같이 명령문의 구별된 부분.

절단. 메모리 용량이나 저장영역 용량을 초과할 때 조작 결과의 일부를 버리는 프로세스.

절대 경로. 오브젝트의 전체 경로 이름입니다. 절대 경로 이름은 최상위 레벨이나, "루트" 디렉토리(슬래시나(/)나 백슬래시(\)에 의해 식별되는)에서 시작합니다.

점검 무결성. OS/390용 DB2 UDB에서, 테이블의 각 행이 그 테이블에 정의된 테이블 점검 제한조건을 준수할 때 존재하는 상태. 점검 무결성을 유지하려면 OS/390용 DB2 UDB에서 데이터를 추가하거나 변경하는 조작에 대해 테이블 점검 제한조건을 시행해야 합니다.

점검 보류 상태. 테이블이 갱신될 때 제한조건이 점검되지 않고 테이블에 대해 제한된 활동만 허용되는 상태로 될 수 있는 테이블 상태.

점검 제한조건. 제한조건이 정의된 테이블의 각 행에 대해 거짓이 아닌 점검 조건을 지정하는 제한조건.

점검 조건. 점검 제한조건에 사용되는 검색 조건의 제한된 형식.

점검점. OS/390용 DB2 UDB가 로그에 내부 상태 정보를 기록하는 지점. 서브시스템이 비정상적으로 종료된 경우 복구 프로세스에서 이 정보를 사용합니다.

접속. DB2에서, 인스턴스 레벨에서 오브젝트에 원격 액세스.

접속 기능. OS/390용 DB2 UDB 및 TSO, IMS™, CICS 또는 일괄처리 주소 공간 사이의 인터페이스. 접속 기능은 응용프로그램이 OS/390용 DB2 UDB에 액세스할 수 있도록 합니다.

정량 술어. 값 집합과 값을 비교하는 술어.

정리. 기존 데이터를 대체하면서, 사용자 테이블에서 관계가 있는 모든 데이터가 목표 테이블로 복사되는 프로세스. **완전 새로 고침** 및 **부분 새로 고침**도 참조.

정리. 데이터 웨어하우스에서 사용할 수 있도록 만들기 위해 운영 시스템에서 추출되는 데이터를 조작하는 프로세스.

정밀도. 숫자 데이터 유형에서, 부호를 제외한 총 2진 또는 십진 자리수.

정보 카탈로그. 사용자가 조직에서 사용할 수 있는 데이터와 정보를 식별하고 찾을 수 있도록 도와주는 설명적 데이터(*비지니스 메타데이터*)가 있는, Information Catalog Manager에 의해 관리되는 데이터베이스. 정보 카탈로그에는 일부 *기술 메타데이터*도 있습니다.

정상 식별자. (1) SQL에서, 뒤에 0개 이상의 문자가 오는 영문자로, 이름을 만들기 위해 사용되는 각 문자는 영문자(a-z 및 A-Z), 기호, 숫자 또는 밑줄입니다. (2) OS/390용 DB2 UDB에서, 뒤에 0개 이상의 문자가 오는 *대문자*로, 각각은 *대문자*, 숫자 또는 밑줄입니다. 일반 식별자는 예약어가 아니어야 합니다.

정상 토큰. 숫자 상수, 일반 식별자, 호스트 식별자 또는 키워드.

정상화. 데이터베이스에서, 가장 간단한 형식으로 관계를 축소시킴으로써 데이터 모델을 재구성하는 프로세스.

정의 메타데이터. Data Warehouse Center에서, 데이터 웨어하우스(스키마), 데이터의 소스 및 데이터 로드시 적용되는 변환의 형식에 대한 정보.

정적 바인드. SQL문이 사전에 컴파일된 후 바인드되는 프로세스. 모든 정적 SQL문은 동시에 실행되도록 준비됩니다. 또한 *바인드* 참조.

정적 SQL. 프로그램 내에 내포되어 있고, 프로그램이 실행되기 전에 프로그램 준비 프로세스 중에 준비되는 SQL문. 준비되고 나면, 명령문에 의해 지정된 호스트 변수의 값이 변경될 수 있어도 정적 SQL문은 변경되지 않습니다.

제어 간격 정의 필드(CIDF). VSAM에서, 각 제어 간격의 끝에 있는 4 바이트의 필드로, 제어 간격의 여유 공간(있으면)을 설명합니다.

제어 간격(CI). VSAM에서, VSAM이 레코드를 저장하고 분산 여유 공간을 작성하는 직접 액세스 저장영역의 고정 길이 영역. 또한, 키순 데이터 세트나 파일에서, 순서 세트 색인 레코드의 항목에 의해 지시되는 레코드 세트. 제어 간격은 VSAM이 직접 액세스 저장영역으로, 또는 그 저장영역으로부터 전송하는 정보의 단위입니다. 제어 간격에는 항상 절대 필요한 수의 물리적 레코드가 포함됩니다.

제어 메타데이터. Data Warehouse Center에서, 테이블이 단계 처리에 의해 갱신되는 날짜 및 시간과 같이, 웨어하우스 변경사항에 대한 정보.

제어 서버. DB2 복제에서, 적용 가능한 복사 작업 내역 정의 및 Apply 프로그램 제어 테이블의 데이터베이스 위치.

제어 센터. 데이터베이스 오브젝트(예: 데이터베이스, 테이블)와 이들 사이의 관계를 표시하는 그래픽 인터페이스. 제어 센터를 통해, DBA 유틸리티, Visual Explain 및 성능 모니터 도구에서 제공되는 타스크를 수행할 수 있습니다. 반의어 - **DataJoiner 복제 관리(DJRA)** 도구.

제어 테이블. DB2 복제에서, 복제 소스와 복사 작업 내역 정의 또는 기타 복제 제어 정보가 저장되는 테이블.

제어권. 오브젝트에 대한 완전한 제어 권한. 여기에는 오브젝트의 액세스, 삭제, 변경 권한과, 그 오브젝트의 특권을 다른 사용자에게 확대시키거나 권한 취소시키는 권한도 포함됩니다.

제어점. (1) APPN에서, 노드의 자원을 관리하고 네트워크의 다른 노드로 서비스를 선택적으로 제공하는 노드 구성요소. 그 예는 유형 5 노드의 시스템 서비스 제어점(SSCP), 유형 4 노드의 물리적 장치 제어점(PUCP), 유형 2.1(T2.1)의 네트워크 노드 제어점(NNCP), T2.1 끝 노드의 끝 노드 제어점(ENCP) 등이 있습니다. SSCP 및 NNCP는 다른 노드로 서비스를 제공할 수 있습니다. (2) 그 노드의 자원을 관리하는 T2.1 노드의 구성요소. T2.1 노드가 APPN 노드인 경우, 제어점은 다른 APPN 노드와 함께 지점간 세션을 제어할 수 있습니다. T2.1 노드가 네트워크 노드인 경우, 제어점도 T2.1 네트워크의 인접 끝 노드로 서비스를 제공합니다. 또한 *물리적 단위* 참조.

제한조건. 테이블에서 삽입, 삭제 또는 갱신될 수 있는 값을 제한하는 규칙. *점검 제한조건*, *참조 제한조건* 및 *고유 제한조건* 참조.

조각. OS/390 환경에서, 비파티션 페이지 세트의 데이터 세트.

조건부 재시작. 사용자 정의 조건부 재시작 제어 레코드(CRCR)에 의해 지시되는 OS/390용 DB2 UDB 재시작.

조인. 일치하는 컬럼 값을 기초로 두 개 이상의 테이블로부터 데이터를 검색할 수 있는 SQL 관계 조작.

조정 가능 잠금. OS/390용 DB2 UDB에서, 경쟁하는 사용자 사이의 협의에 의해 해당 모드 등급을 낮춰서 모두에게 호환되도록 하는 잠금. 물리적 잠금은 조정 가능 잠금의 예입니다.

조정 매개변수 테이블. Capture 프로그램에 의해 사용되는 타이밍 정보가 들어 있는 소스 서버에서의 테이블. 이 정보에는 다음이 포함됩니다.

- 데이터 변경 테이블에서의 행 보유 기간.
- 데이터베이스 로그나 저널에 변경사항이 저장되기 전의 경과 시간.
- 변경된 데이터를 작업 테이블 단위로 확약시키는 빈도.

조정 에이전트. 응용프로그램으로부터 데이터베이스 관리 프로그램에 의해 요청이 수신될 때 시작되는 에이전트. 이는 응용프로그램이 지속되는 동안 응용프로그램과 연관된 상태로 남아 있습니다. 이 에이전트는 응용프로그램에 대해 작동되는 부속 에이전트들을 조정합니다. 또한 *부속에이전트* 참조.

조정자. *자원 한계 기능* 참조.

조정자. OS/390용 DB2 UDB에서, 하나 이상의 다른 시스템에서 수행되는 작업을 포함하는 작업 단위의 확약 또는 구간 복원을 조정하는 시스템 구성요소.

조정자 노드. 응용프로그램이 원래 연결되어 있고 조정 에이전트가 상주하는 노드.

조정자 부속 절. 다른 부속 절(이 있는 경우)을 시작하고 그 결과를 응용프로그램에 리턴하는 응용프로그램의 부속 절.

조합 순서. 색인 데이터를 순서대로 정렬, 병합, 비교 및 처리하기 위한 목적으로 되어 있는 문자의 순서.

조합 조인. 다음 조건들이 만족될 때 조인되는 두 테이블의 결과.

- 테이블들이 같은 데이터베이스 파티션에서 단일 파티션 노드 그룹에 상주하거나, 같은 파티션 노드 그룹에 있으면서 파티션 컬럼 수가 같고, 컬럼들이 파티션과 호환되며, 두테이블 모두 같은 파티션 함수를 사용합니다.
- 모든 파티션 키 컬럼 쌍들은 equijoin 술어에 참여합니다.

조회. (1) 특정 조건을 기초로 데이터베이스로의 정보를 원하는 요청으로, 예를 들어, 고객 테이블에서 잔액이 $1000보다 모든 고객 목록을 원하는 요청. (2) OS/390용 DB2 UDB에서, 결과 테이블을 지정하는 특정 SQL문의 구성요소.

조회 내 병렬 처리. 파티션 내 병렬 처리, 파티션간 병렬 처리 또는 둘 다를 사용하여 동시에 단일 조회의 부분들을 처리할 수 있는 능력.

조회 블록. OS/390용 DB2 UDB에서, FROM 절 중 하나에 의해 표시되는 조회의 부분. 각 FROM 절에는 OS/390용 DB2 UDB가 내부적으로 조회를 처리하는 방법에 따라 여러 조회 블록이 있을 수 있습니다.

조회 CP 병렬 처리. OS/390용 DB2 UDB에서, 여러 타스크를 사용하여 처리되는 단일 조회의 병렬 실행. 반의어 - *Sysplex 조회 병렬 처리.*

조회 I/O 병렬 처리. OS/390용 DB2 UDB에서, 단일 조회 내의 여러 입출력 요청을 트리거하여 처리되는 데이터의 병렬 액세스.

종속. SQL에서, 최소한 하나의 상위 요소를 가지고 있는 오브젝트(행, 테이블 또는 테이블 공간). *상위 행*, *상위 테이블*, *상위 테이블 공간* 참조.

종속 논리 장치(DLU). LU간 세션을 인스턴스화하기 위해 시스템 서비스 제어점(SSCP)으로부터의 보조가 요구되는 논리 장치.

종속 에이전트. *서브에이전트* 참조.

종속 테이블. 적어도 하나의 참조 제한조건이 있는 종속인 테이블.

종속 행. 상위 행에서 상위 키의 값과 일치하는 외부 키를 포함하는 행. 외부 키 값은 종속 행에서 상위 행으로의 참조를 나타냅니다.

주제 영역. Data Warehouse Center에서, 특정 논리 비지니스 영역에 대한 웨어하우스 데이터를 작성하는 프로세스 세트. 주제 영역의 프로세스는 특정 오브젝트에서 필요로 하는 세부사항 데이터, 데이터 요약 및 큐브를 작성하기 위해 그 오브젝트에서 작동됩니다.

준비된 SQL문. SQL에서, PREPARE문에 의해 처리된 SQL문의 실행 양식인 명명된 오브젝트.

중간 네트워크 노드. APPN에서, 원래의 논리 장치(OLU) 및 목적지 논리 장치(DLU) 사이의 라우트 부분인 노드.

중첩 테이블 표현식. (1) FROM절에 지정된 fullselect의 평가를 통해 하나 이상의 다른 테이블로부터 직간접적으로 얻게 되는 결과 테이블. (2) OS/390용 DB2 UDB에서, FROM 절의 부속 선택(괄호로 묶인).

증분식 바인드. SQL문이 바인드 프로세스 동안 바인드될 수 없지만 VALIDATE(RUN)가 지정되었으므로, 응용프로그램 프로세스 실행 동안 그 SQL문이 바인드되는 프로세스. 또한 *바인드* 참조.

지속기간. SQL에서, 시간 간격을 나타내는 숫자. *날짜 지속기간*, *레이블된 지속기간* 및 *시간 지속기간* 참조.

지속성. Net.Data에서, 트랜잭션이 여러 Net.Data 호출을 분산되어 있을 때 전체 트랜잭션에 대해 지정된 값을 보존하는 상태. 변수만 지속될 수 있습니다. 또한, 확약 제어의 영향을 받는 자원에 대한 조작은 명시 확약이나 구간 복원이 수행될 때까지, 또는 트랜잭션이 완료될 때 활동중 상태를 보존합니다.

지역. 데이터베이스 관리 프로그램에 의한 내부 처리를 위해 국가 코드에 맵핑되는 POSIX 로케일의 부분.

지역. 지역 서브시스템이 유지보수하는 오브젝트를 가리키될 사용되는 용어. 예를 들어, OS/390용 DB2 UDB에서 지역 테이블은 지역 DB2 서브시스템에 의해 유지보수되는 테이블입니다. 반의어 - *원격*.

지역 갱신. 사본이 아닌 기본 테이블에 대한 갱신.

지역 데이터베이스. 물리적으로 사용중인 워크스테이션에 위치한 데이터베이스. 반의어 - *원격 데이터베이스*.

지역 데이터베이스 디렉토리. 데이터베이스가 물리적으로 상주하는 디렉토리. 지역 데이터베이스 디렉토리에 표시되는 데이터베이스들은 시스템 데이터베이스 디렉토리와 같은 노드상에 위치합니다.

지역 서브시스템. 사용자나 응용프로그램이 직접 연결되는(OS/390용 DB2 UDB의 경우, OS/390용 DB2 UDB 접속 기능 중 하나에 의해) 고유한 RDBMS.

지역 잠금. DB2간 동시처리 제어를 제공하지만, DB2간 동시처리 제어는 제공하지 않는 잠금으로, 그 범위는 단일 OS/390용 DB2 UDB 시스템입니다.

지역 테이블 잠금. 단일 데이터베이스 파티션에서만 얻는 테이블 잠금.

차

차원. OLAP Starter 킷에서, 시간, 계정, 제품 또는 판매와 같은 데이터 범주. 차원은 다차원 데이터베이스 윤곽에서 최상위 조정 레벨을 나타냅니다.

참조 구조. OS/390용 DB2 UDB에서, 최소하나의 테이블을 포함하고 있는 테이블 및 관계 세트, 그리고 세트의 모든 테이블에 대해, 테이블이 참여하는 모든 관계와 그 테이블이 관련되는 모든 테이블.

참조 무결성. (1) 모든 외부 키 값이 유효한 데이터베이스 상태. (2) 테이블의 한 컬럼에 있는 데이터로부터 같거나 다른 테이블의 또 다른 컬럼에 있는 데이터로의 의도한 모든 참조사항이 유효할 경우에 존재하는 상태. 참조 무결성을 유지하려면 OS/390용 DB2 UDB에서 모든 LOAD, RECOVER, INSERT, UPDATE, DELETE 조작에 대한 참조 제한조건을 시행해야 합니다.

참조 제한조건. 상위 키 값으로서 나타나는 경우에만 외부 키의 널(NULL) 아닌 값이 유효한 참조 무결성 규칙.

첫번째 실패 서비스 로그. 진단 메시지, 진단 데이터, 경보 정보 및 관련된 덤프 정보를 포함하는 파일(db2diag.log). 이 파일은 데이터베이스 관리자에 의해 사용됩니다.

청구. OS/390용 DB2 UDB에서, 오브젝트가 액세스되는 DBMS로의 통지. 청구는 해제될 때까지(보통 확약 지점에서 발생) 드레인의 발생을 방지합니다. *드레인*도 참조.

청구 계수. OS/390용 DB2 UDB에서, 오브젝트에 액세스하는 에이전트 수.

청구 클래스. OS/390용 DB2 UDB에서, 커서 안정성(CS), 반복 읽기(RR), 쓰기 유형 중 하나가 될 수 있는 측정의 오브젝트 액세스 유형.

초기화 fullselect. 소스 테이블로부터 초기 값의 직접 하위를 얻는 순환 공통 테이블 표현식에서의 첫번째 fullselect.

총계 함수. *컬럼 함수*의 동의어.

최신 로그 레코드. 사용중인 로그에 가장 최근 기록된 로그 레코드.

최적화 알고리즘. 많은 대체 액세스 플랜의 실행 비용을 모델링하고 계산된 비용이 최소인 것을 선택하여 DML 명령문에 대한 액세스 플랜을 선택하는 SQL 컴파일러의 구성요소.

최적화 SQL 텍스트. 액세스 플랜을 선택하기 위해 최적화 알고리즘에서 실제로 사용되는 조회를 기초로 하는 Explain 기능에서 생성되는 SQL 텍스트. 이 조회는 명령문 컴파일 동안 SQL 컴파일러의 다양한 구성요소에 의해 보충되어 다시 작성됩니다. 텍스트는 내부 표시로부터 재구성되어 원해 SQL 텍스트와 다르게 됩니다. 최적화된 명령문은 원래 명령문과 동일한 결과를 산출합니다.

추적. OS/390용 DB2 UDB 모니터링, 감사, 성능, 계정, 통계 및 서비스 가능성(전역) 데이터를 모니터하고 수집할 수 있는 능력을 제공하는 OS/390용 DB2 UDB 기능.

출력 파일. 레코드 작성을 허용하는 옵션으로 열리는 장치 파일 또는 데이터베이스.

충돌 감지. update-anywhere 복제 구성에서:

- 제한조건 오류를 검출하는 프로세스.
- 같은 복제 주기 동안 소스와 목표 테이블에서 같은 행이 갱신된 경우에 검출하는 프로세스. 충돌이 감지되면 충돌을 야기시킨 트랜잭션이 거부됩니다. 또한 *확장 충돌 검출*, *표준 충돌 검출* 및 *행 복제 충돌 검출* 참조.

카

카탈로그. 데이터베이스 관리 프로그램에 의해 유지보수되는 테이블 및 뷰의 집합. 이 테이블 및 뷰에는 테이블, 뷰 및 색인의 설명과 같이, 정보에 대한 정보가 있습니다.

카탈로그 노드. 카탈로그 테이블이 상주하는 노드. 카탈로그 노드는 각 데이터베이스에 대해 다른 노드가 될 수 있습니다.

카탈로그 뷰. 관리를 위해 Text Extender에서 작성되는 시스템 테이블의 뷰. 카탈로그 뷰에는 Text Extender에서 사용할 수 있는 테이블과 컬럼에 대한 정보가 있습니다.

카탈로그 테이블. OS/390용 DB2 UDB 카탈로그의 테이블.

캐쉬. 자주 액세스되는 지시문 및 데이터가 들어 있는 버퍼. 캐쉬를 사용하여 액세스 시간을 줄일 수 있습니다.

캐쉬 관리 프로그램. Net.Data®에서, 한 워크스테이션에 대한 캐쉬를 관리하는 프로그램. 캐쉬 관리 프로그램은 여러 캐쉬를 관리할 수 있습니다.

캐쉬 구조. 병렬 Sysplex®의 모든 구성원이 사용할 수 있는 데이터를 저장하는 결합 기능 구조. OS/390용 DB2 UDB 데이터 공유 그룹은 그룹 버퍼 풀로서 캐쉬 구조를 사용합니다.

캐싱. 정보를 새로 고쳐야 하는 시간이 될 때까지, 빠르게 검색할 수 있도록 요청에서 자주 사용되는 결과를 웹 서버에 지역적으로 저장하는 프로세스.

캡쳐 트리거. DB2 복제에서, IBM 소스 테이블이 아닌 다른 소스 테이블에서 수행되는 조작을 캡쳐, 삭제 갱시 및 삽입하는 메카니즘. 반의어 - *캡쳐 프로그램* 및 *프로그램 적용*.

커서. 일부 순서화된 행 집합 내부에 있는 특정 행을 가리키기 위해 응용프로그램이 사용하는 명명된 제어 구조. 커서는 집합에서 행을 검색하는 데 사용됩니다.

커서 안정성(CS). 커서가 행에 있는 동안 응용프로그램 트랜잭션에 의해 액세스되는 행을 잠그는 분리 레벨. 다음 행이 페치되거나 트랜잭션이 종료될 때까지 잠금 상태가 유지됩니다. 행에 있는 데이터가 변경되는 경우, 데이터베이스로 변경사항이 확약될 때까지 잠금이 유지됩니다.

커서 테이블(CT). OS/390용 DB2 UDB에서, 실행 중인 응용프로그램 프로세스에서 사용되는 윤곽 커서의 사본.

컨테이너. *테이블 공간 컨테이너* 참조.

컬럼 분배 값. 일부 컬럼 또는 quantile 값 중에서 가장 빈도가 잦은 값들을 설명하는 통계. 이 값들은 최고의 액세스 플랜을 결정하는 것을 돕기 위해 최적화 알고리즘에 사용됩니다.

컬럼 함수. (1) 여러 행들의 값에 적용되는 조회에서 사용되는 연산. 컬럼 함수에는 SUM, AVG, MIN, MAX, COUNT, STDDEV, VARIANCE 등이 포함됩니다. 동의어 - *집계 함수*. (2) OS/390용 DB2 UDB에서, 하나 이상의 행에서 값 콜렉션으로부터 결과를 파생시키는 SQL 조작. 반의어 - *스칼라 함수*.

코드 세트. 시스템과 입출력 장치 사이의 인터페이스를 제공하는 문자 세트에 대한 코드화 값. ISO는 IBM이 정의한 용어 코드 페이지에 상응하는 용어로서 코드 세트를 사용합니다.

코드 페이지. 코드 값이 지정된 문자 집합.

코드 포인트. CDRA에서, 코드 페이지 내 문자를 나타내는 고유한 비트 패턴.

코드화 문자 세트. 세트 문자와 코드화 표시 사이의 일대일 관계 및 문자 세트를 설정하는 결정 규칙 세트.

코드화 문자 세트 식별자(CCSID). 코드화 그래픽 문자 표시를 고유하게 식별하는 코드화 체계 식별자, 문자 세트 식별자, 코드 페이지 식별자 등의 정보를 포함하는 숫자.

코드화 체계. 문자 데이터를 표시하는 규칙 집합.

콜 레벨 인터페이스(CLI). Embedded SQL API 대신 사용되는 데이터베이스 액세스에 대한 호출가능 API. Embedded SQL과 대조해 볼 때, CLI에서는 사용자의 사전 컴파일링이나 바인딩을 요구하지 않지만, 런타임 시 SQL문과 이에 관련되는 서비스를 처리하기 위한 표준 함수 세트를 제공합니다.

콜드 스타트. (1) 초기 프로그램 로드 프로시듀어를 사용하여 시스템이나 프로그램을 시작하는 프로세스. 반의어 - *웜 시동*. (2) OS/390용 DB2 UDB가 로그 레코드 처리 없이 재시작하는 프로세스.

콜렉션. OS/390용 DB2 UDB에서, 같은 규정자가 있는 패키지 그룹.

크래쉬 복구. 바로 전의 실패로부터 복구하는 프로세스.

클라이언트. (1) 데이터베이스 서버와 통신하고 액세스하는 프로그램(또는 수행 중인 워크스테이션). (2) *리퀘스터* 참조.

클러스터 색인. 연속된 키 값이 테이블에 저장된 행의 연속과 일치하는 색인. 일치 정도는 최적화 알고리즘에 의해 사용되는 통계에 의해 측정됩니다.

클리에트. 웹 서버로부터의 요청에 대한 서비스를 제공하는 Net.Data Live Connection에서의 장기 수행 프로세서. 연결 관리 프로그램은 클리에트 프로세스의 스케줄을 정하여 이러한 요청에 대해 서비스를 제공합니다.

키. 테이블, 색인 또는 참조 제한조건 설명에서 식별되는 컬럼의 순서화된 콜렉션 또는 컬럼.

키 값에 기초한 파티션 전략. 데이터베이스 파티션으로 테이블에 있는 행들을 지정하는 전략. 행들은 파티션 키 컬럼 값들에 기초하여 지정됩니다.

키순 데이터 세트(KSDS). OS/390 환경에서, 레코드가 키순으로 로드되고 색인별로 제어되는 VSAM 파일 또는 데이터 세트.

키워드. (1) 컴퓨터, 명령어 또는 응용프로그램의 사전 정의된 단어 중 하나. (2) SQL문에서 사용되는 옵션을 식별하는 이름.

타

타스크 제어 블록(TCB). OS/390용 DB2 UDB에 연결되어 있는 주소 공간 내에서 타스크에 대한 정보를 통신하기 위해 사용되는 제어 블록. 하나의 주소 공간이 많은 타스크 연결(하나의 타스크당 많이)을 지원하지만, 주소 공간 연결은 하나만 지원됩니다.

타스크의 비정상 종료(이상 종료). OS/390용 DB2 UDB에서, 실행 중 복구 기능을 분석할 수 있는 오류 상태로 인해 타스크, 작업 또는 서브시스템이 종료되었습니다.

테이블. 일부 순서화되지 않은 행과 특정 수의 컬럼으로 구성되는 명명된 데이터 오브젝트. 또한 *기본 테이블* 참조.

테이블 공간. (1) 데이터베이스 오브젝트가 저장되는 컨테이너 콜렉션을 추출한 것. 테이블 공간은 데이터베이스 내에 저장된 테이블과 데이터베이스 사이의 간접 레벨을 제공합니다. 테이블 공간에는,

- 이 공간에 지정된 미디어 저장영역 장치 공간이 있습니다.
- 그 안에 작성된 테이블이 있습니다. 이 테이블들은 테이블 공간에 속하는 컨테이너에서 공간을 사용합니다. 테이블의 LOB 부분, 긴 필드, 색인, 데이터는 동일한 테이블 공간에 저장될 수 있고, 별도의 테이블공간으로 개별적으로 분리시킬 수도 있습니다.

(2) OS/390용 DB2 UDB에서, 하나 이상의 테이블에서 레코드를 저장하기 위해 사용되는 페이지 세트.

테이블 공간 세트. OS/390용 DB2 UDB에서, 다음 이유 중 하나로 함께 복구되어야 하는 테이블 공간 및 파티션 세트.

- 각 테이블 공간 및 파티션에 다른 것 중 하나에 있는 테이블에 대해 상위이거나 종속적인 테이블이 포함되어 있습니다.
- 세트에 기본 테이블 및 연관된 보조 테이블이 있습니다.

테이블 공간에 두 유형의 관계가 포함될 수 있습니다.

테이블 공간 컨테이너. 테이블 공간으로의 공간 할당을 나타내는 일반 용어. 테이블 공간 유형에 따라 컨테이너는 디렉토리, 장치 또는 파일이 될 수 있습니다.

테이블 대기행렬. 데이터베이스 노드 사이에 행을 전송하는 메카니즘. 테이블 대기행렬은 간단한 행 삽입 및 제거 규칙으로 행렬이 분배됩니다. 테이블 대기행렬은 일련의 데이터베이스에서 서로 다른 프로세스 사이에 행을 전달하는 데에도 사용할 수 있습니다.

테이블 위치 지정자. OS/390용 DB2 UDB에서, SELECT문의 FROM 절이나 INSERT문의 부속 선택에서, 또는 사용자 정의 함수 내에서 전이 테이블을 트리거하기 위한 액세스를 허용하는 메카니즘. 테이블 위치 지정자는 전이 테이블을 나타내는 모든 단어 정수 값입니다.

테이블 점검 제한조건. OS/390용 DB2 UDB에서, 기본 테이블의 특정 컬럼에 포함할 수 있는 값을 지정하는 사용자 정의 제한조건.

테이블 지정자. 특정 오브젝트 테이블을 지시하는 컬럼 이름 규정자.

테이블 함수. OS/390용 DB2 UDB에서, 인수 세트를 수신하고 함수를 참조하는 SQL문에 테이블을 리턴하는 함수. 테이블 함수는 부속 선택의 FROM 절에서만 참조될 수 있습니다.

토큰. 전산 언어의 기본적인 구문 단위. 토큰은 공백 문자와, 문자열 상수나 분리 식별자 내의 문자를 제외하고 하나 이상의 문자들로 구성됩니다.

토폴로지 및 라우팅 서비스(TRS). 토폴로지 데이터베이스를 관리하고 경로를 계산하는 APPN 제어점 구성요소.

통과. 연합 데이터베이스 시스템에서, 사용자가 데이터 소스의 SQL SQL dialect에서 데이터 소스와 통신할 수 있는 기능.

통신 데이터베이스(CDB). 원격 데이터베이스 관리 시스템과의 대화를 설정하기 위해 사용되는 OS/390용 DB2 UDB 카탈로그의 테이블 세트.

통지 프로세스. 한 단계가 완료될 때 통지를 위해 작성되는 모든 단계를 포함하는, Data Warehouse Center에 의해 작성되는 프로세스.

트랜잭션. (1) 특정 조치 또는 결과를 수행하는 두 프로그램이나 두 워크스테이션 또는 프로그램과 하나의 워크스테이션 사이의 교환. 고객 수지 균형의 갱신과 고객 에치금 항목이 그 예입니다. 동의어 - *작업 단위(UOW)*. (2) 하나의 Net.Data 호출. 지속적인 Net.Data가 사용될 경우, 트랜잭션은 여러 Net.Data 호출에 분산될 수 있습니다.

트랜잭션 관리자. 트랜잭션에 식별자를 지정하고, 진행을 모니터하며, 트랜잭션 완료 및 실패 복구를 수행하는 함수.

트랜잭션 관리자 데이터베이스(TM 데이터베이스). DB2 데이터베이스와 함께 두 단계 확약(SYNCPOINT TWOPHASE)이 사용될 때 트랜잭션을 로그하는 데 사용되는 데이터베이스. 트랜잭션 실패시, TM Database 정보에 액세스하여 실패한 트랜잭션에 관련된 데이터베이스들을 다시 동기화할 수 있습니다.

트랜잭션 보상. 거부되는 확약 트랜잭션에 의해 영향받은 행을 복원하는 프로세스. 확약된 트랜잭션이 거부될 때, 트랜잭션이 확약되기 전에 있던 상태로 행이 복원됩니다.

트랜잭션 잠금. OS/390용 DB2 UDB에서, SQL문의 동시 실행을 제어하기 위해 사용되는 잠금.

트랜잭션 프로그램 이름. SNA LU 6.2 대화에서, 대화의 다른 반쪽이 되는 원격 논리 장치에 있는 프로그램의 이름.

트랜잭션 프로그램(TP). 상대 응용프로그램과 통신하기 위해 APPC를 사용하는 응용프로그램.

트리거. (1) DB2에서, 특정 SQL문이 수행될 때 데이터베이스 관리 프로그램에 의해 간접 호출되는 데이터베이스의 오브젝트. (2) OS/390용 DB2 UDB 데이터베이스에 저장되고 OS/390용 DB2 UDB 테이블에서 특정 이벤트가 발생할 때 실행되는 SQL문 세트.

트리거 내용. OS/390용 DB2 UDB에서, 트리거가 활성화되고 해당되는 트리거 조치 조건이 참으로 평가될 경우에 실행되는 SQL문 세트.

트리거 삭제. OS/390용 DB2 UDB에서, 트리거링 SQL 조작 DELETE로 정의된 트리거.

트리거 수준. OS/390용 DB2 UDB에서, 트리거가 활성화되는 지를 판별하는 트리거 특성.
- 트리거 SQL문마다 한번씩만.
- SQL문이 수정하는 각 행마다 한번씩.

트리거 연쇄. OS/390용 DB2 UDB에서, 트리거의 트리거 조치로 다른 트리거가 활성화될 경우에 발생하는 프로세스.

트리거 이벤트. OS/390용 DB2 UDB에서, 트리거 정의에서 해당 트리거의 활성화를 야기하는 지정된 조작. 트리거 이벤트는 트리거링 조작(INSERT, UPDATE 또는 DELETE)과 조작이 수행되는 트리거링 테이블로 구성됩니다.

트리거 이벤트. 트리거 정의에서, 트리거가 수행되도록 하는 갱신 조작(INSERT, UPDATE 또는 DELETE문).

트리거 조치. (1) 트리거 이벤트가 발생할 때 실행되는 조치. (2) OS/390용 DB2 UDB에서, 트리거가 활성화될 때 수행되는 SQL 논리. 트리거 조치는 선택적인 트리거 조치 조건과, 조건이 참으로 평가될 경우에만 실행되는 트리거 SQL문 세트로 구성됩니다.

트리거 조치 조건. (1) 트리거 조치 내에서의 SQL문 실행을 제어하는 검색 조건. (2) OS/390용 DB2 UDB에서, 트리거 조치의 선택적 부분. 이러한 부울 조건은 WHEN 절로 표시되며 트리거 SQL문이 실행되어야 하는 지를 판별하기 위해 DB2가 평가하는 조건을 지정합니다.

트리거 패키지. OS/390용 DB2 UDB에서, CREATE TRIGGER문이 실행될 때 작성되는 패키지. 패키지는 트리거가 활성화될 때 실행됩니다.

트리거 활성화. OS/390용 DB2 UDB에서, 트리거 정의에 정의된 트리거 이벤트가 실행될 때 발생하는 프로세스. 트리거 활성화는 트리거 조치 조건의 평가와 트리거 SQL문의 조건부 실행으로 구성됩니다.

트리거 활성화 시간. OS/390용 DB2 UDB에서, 트리거 이벤트 이전이나 이후에 트리거가 활성화되어야 하는 지에 대한 트리거 정의에서의 표시.

트리거 SQL문. OS/390용 DB2 UDB에서, 트리거가 활성화되고 해당되는 트리거 조치 조건이 참으로 평가될 경우에 실행되는 SQL문 세트. 트리거 SQL문은 *트리거 내용*이라고도 합니다.

트리거링 테이블. OS/390용 DB2 UDB에서, 트리거가 작성되는 테이블. 정의된 트리거 이벤트가 이 테이블에서 발생할 경우, 트리거가 활성화됩니다.

트리거링 SQL 조작. OS/390용 DB2 UDB에서, 트리거링 테이블에서 수행될 때 트리거의 활성화를 야기하는 SQL 조작.

특권. (1) 특정 방식으로 특정 데이터베이스 오브젝트에 액세스하는 권리. 이 권한은 오브젝트 작성자 또는 DBADM(데이터베이스 관리자) 권한이나 SYSADM(권한 관리자) 권한을 가진 사용자에 의해 제어됩니다. 특권에는 테이블로부터 데이터 작성, 삭제, 선택과 같은 권한이 포함됩니다. (2) OS/390용 DB2 UDB에서, 간혹 특정 오브젝트에서 특정 함수를 수행하는 능력. *명시적 특권* 및 *내재된 특권*도 참조.

특권 세트. 설치 SYSADM ID의 경우, 가능한 모든 특권의 세트. 다른 권한 부여 ID의 경우, OS/390용 DB2 UDB 카탈로그에 그 ID에 대해 기록되는 모든 특권 세트.

특수 레지스터. 데이터베이스 관리 프로그램에 의해 응용프로그램 프로세스에 대해 정의되어 있고, SQL문에서 참조될 수 있는 정보를 저장하는 데 사용되는 저장영역. 그 예는 USER와 CURRENT DATE입니다.

파

파일 서버. NetWare 운영 체제 소프트웨어를 실행하고 네트워크 서버로 작동하는 워크스테이션. DB2는 파일 서버를 사용하여 DB2 서버 주소 정보를 저장합니다. 이 정보는 DB2 클라이언트가 검색하여 IPX/SPX 클라이언트-서버 연결을 설정하는 정보입니다.

파일 입력을 사용하는 SQL 프로세서(SPUFI). OS/390용 DB2 UDB에서, 파일 입력을 사용하는 SQL 프로세서. DB2I 사용자가 응용프로그램에 SQL문을 삽입하지 않아도 그 SQL문을 실행할 수 있게 하는 TSO 접속 부속 구성요소의 기능.

파일 참조 변수. 클라이언트 메모리 버퍼에서가 아닌 클라이언트상의 파일에 데이터가 상주함을 나타내는 데 사용되는 호스트 변수.

파티션. OS/390 환경에서, 페이지 세트의 부분. 각 파티션은 독립적으로 호가장 가능한 단일 데이터 세트에 해당됩니다. 파티션은 파티션된 페이지 세트의 파티션 수에 따라 최대 크기 1, 2 또는 4 GB까지 확장될 수 있습니다. 제공된 페이지 세트에 대한 모든 파티션의 최대 크기는 같습니다.

파티션 내 병렬 처리. 단일 데이터베이스 파티션 내에서 같은 시간에 여러 데이터베이스 조작(예: 색인 작성, 데이터베이스 로드 및 SQL 조회)을 수행할 수 있는 능력. 반의어 - *파티션간 병렬 처리*.

파티션 맵. 파티션 맵 색인을 노드 그룹에 있는 데이터베이스 파티션으로 맵핑하는 파티션 번호의 벡터.

파티션 맵 색인. 해쉬 파티션 또는 범위 파티션으로 지정되는 번호.

파티션 키. (1) 해당 테이블에 있는 순서화된 하나 이상의 컬럼 세트. 테이블의 각 행에 대해 파티션 키 컬럼의 값들은 행이 속해 있는 데이터베이스 파티션이 어떤 것인지를 결정하는 데 사용됩니다. (2) 복제에서, 해당 테이블에 있는 하나 이상의 순서화된 컬럼 세트. 소스 테이블의 각 행에 대해, 파티션 키 컬럼의 값들은 행이 어떤 목표 테이블에 속해 있는지 판별하는 데 사용됩니다.

파티션 테이블 공간. OS/390 환경에서, 색인 키 범위를 기초로 부분들로 다시 세분화되는 테이블 공간으로, 각 부분들은 유틸리티에 의해 독립적으로 처리될 수 있습니다.

파티션 호환 조인. 조인되는 모든 행들이 동일한 데이터베이스 파티션에 상주하는 조인.

파티션간 병렬 처리. 파티션 데이터베이스의 여러 파티션에서 같은 시간에 여러 데이터베이스 조작(예: 색인 작성, 데이터베이스 로드 및 SQL 조회)을 수행할 수 있는 능력. 반의어 - *파티션 내 병렬 처리*.

파티션된 데이터 세트(PDS). OS/390 환경에서, 구성원이라고 하는 파티션들로 구분되는 직접 액세스 저장영역의 데이터 세트. 각 파티션에는 프로그램, 프로그램의 부분 또는 데이터가 포함될 수 있습니다. 동의어 - *프로그램 라이브러리*.

파티션된 데이터베이스. 두 개 이상의 데이터베이스 파티션이 있는 데이터베이스. 사용자 테이블 데이터는 하나 이상의 데이터베이스 파티션에 위치할 수 있습니다. 테이블이 여러 파티션상에 있을 때, 한 파티션에 일부 행이 저장되고 다른 파티션에 다른 행들이 저장됩니다. *데이터베이스 파티션*에서 참조하십시오.

파티션된 페이지 세트. OS/390 환경에서, 파티션 테이블 공간이나 색인 공간. 헤더 페이지, 공간 맵 페이지, 데이터 페이지 및 색인 페이지는 파티션 범위 내의 데이터만을 참조합니다.

파티션된 함수. 입력으로서 행의 파티션 키 값을, 출력으로서 파티션 번호를 산출하는 함수.

패널. OS/390용 DB2 UDB에서, 표시 영역(예: 메뉴 패널)에 있는 표시 필드의 표시 위치와 특성을 정의하는 사전 정의된 표시 이미지.

패키지. SQL문 실행에 사용되는 프로그램 준비 중에 산출되는 제어 구조.

패키지 목록. OS/390용 DB2 UDB에서, 응용프로그램 플랜을 확장하기 위해 사용할 수 있는 순서화된 패키지 이름 목록.

패키지 이름. OS/390용 DB2 UDB에서, BIND PACKAGE 또는 REBIND PACKAGE 명령에 의해 작성되는 오브젝트의 이름. 오브젝트는 데이터베이스 요청 모듈(DBRM)의 바인드된 버전입니다. 이름은 위치 이름, 콜렉션 ID, 패키지 ID 및 버전 ID로 구성됩니다.

패킷. 데이터 통신에서, 복합체로서 전송 및 전환되는 2진 숫자의 연속으로서, 데이터와 제어 신호가 포함됩니다.

페이지. (1) *크기*가 4096 바이트(4 KB)인 테이블이나 색인 내의 저장영역 블록. (2) OS/390용 DB2 UDB에서, 테이블 공간(4 KB, 8 KB, 16 KB 또는 32 KB) 또는 색인 공간(4 KB) 내의 저장영역 단위. 테이블 공간에서, 한 페이지에는 한 테이블의 여러 행이 포함됩니다. LOB 테이블 공간에서, LOB 값은 여러 페이지에 분산될 수 있지만 여러 개의 LOB 값이 한 페이지에 저장되지는 않습니다.

페이지 세트. OS/390 환경에서, 테이블 공간이나 색인 공간을 언급하는 또 다른 방식. 각 페이지 세트는 VSAM 데이터 세트 콜렉션으로 구성됩니다.

페이지 세트 복구 보류(PSRCP). OS/390용 DB2 UDB에서, 전체 페이지 세트가 복구되어야 하는 색인 공간의 제한된 상태. 논리 부분의 복구는 허용되지 않습니다.

포워드 복구. 데이터베이스 또는 테이블 공간을 롤 포워드하는 데 사용되는 프로세스. 데이터베이스 로그에 기록된 변경사항을 적용함으로써 지정된 시점으로 복원된 데이터베이스나 테이블 공간이 재작성될 수 있도록 합니다.

표시 이름. 상관 이름이 지정되지 않은 FROM절에 지정된 상관 이름, 테이블 또는 뷰 이름.

표시기 변수. 응용프로그램에서의 널(NULL) 값을 나타내는 데 사용되는 변수. 선택된 컬럼의 값이 널(NULL)인 경우, 음수 값이 표시기 변수에 놓입니다.

표시기 컬럼. OS/390용 DB2 UDB에서, LOB 컬럼 대신 기본 테이블에 저장되는 4 바이트 값.

표준 충돌 감지. Apply 프로그램이 Replica나 사용자 테이블의 변경 테이블에서 이미 캡쳐된 행에서 충돌을 검색하는 *충돌 검출*. *충돌 검출*, *강화된 충돌 검출* 및 *행 Replica 충돌 검출*도 참조.

표현식. 하나의 값을 생성하는 SQL 피연산자나 연산자 및 피연산자의 콜렉션.

프로세스. (1) Data Warehouse Center에서, 원래 양식에서 결정 지원을 위한 양식으로 데이터를 변경하는 소스 데이터에 대해 공통으로 작동되는 일련의 단계들. Data Warehouse Center 프로세스는 하나 이상의 소스, 하나 이상의 단계 및 하나 이상의 목표로 구성됩니다. (2) OS/390용 DB2 UDB에서, OS/390용 DB2 UDB가 자원 및 잠금을 할당하는 단위. 프로세스에는 하나 이상의 프로그램 실행이 포함됩니다. SQL문의 실행은 항상 일부 프로세스와 연관됩니다. 프로세스를 초기화하고 종료하는 수단은 환경에 따라 다릅니다. 동의어 - *응용프로그램 프로세스*.

프로세스간 통신(IPC). 프로세스간에 서로 통신할 수 있도록 하는 운영 체제 기제.

프로시듀어. *저장 프로시듀어* 참조.

프리페치. 사용 전에, 즉 사용을 앞두고 데이터를 읽는 것.

플래거. 선택된 유효성 확인 기준(예: ISO/ANSI SQL92 입력 레벨 표준)을 따르지 않는 응용프로그램에서 SQL문을 식별하는 사전 처리 컴파일러 옵션.

플랜. *응용프로그램 플랜* 참조.

플랜 세그먼트화. OS/390용 DB2 UDB에서, 각 플랜을 섹션들로 나누는 것. 섹션이 필요할 경우, 그 섹션은 독립적으로 EDM 풀에 가져옵니다.

플랜 이름. OS/390용 DB2 UDB에서, 응용프로그램 플랜 이름.

플랜 할당. 실행하기 위해 준비 시 플랜에 OS/390용 DB2 UDB 자원을 할당하는 프로세스.

플랫 파일 인터페이스. 일반 텍스트 파일에서 데이터를 읽고 기록할 수 있도록 하는 Net.Data 내장 함수 세트.

피어 투 피어 통신. 호스트에 의해 관리되지 않는 두 개의 SNA 논리 장치(LU) 사이의 통신. 일반적으로 LU 6.2 모드를 참조할 때 사용됩니다.

피연산자. 연산이 수행되는 엔티티.

필드 프로시듀어. OS/390용 DB2 UDB에서, 단일 값을 수신하여 사용자가 지정할 수 있는 방법으로 이를 전송(암호화 또는 암호해독)하도록 지시된 사용자 작성 exit 루틴.

필터 인수. OS/390용 DB2 UDB에서, 0과, 술어가 참인 테이블의 행 부분을 측정하는 것 사이의 숫자.

하

하위. 오브젝트에 종속되거나 오브젝트의 하위 오브젝트에 종속되는 오브젝트.

하위 네트워킹 노드(LEN 노드). 독립 LU 프로토콜을 지원하지만 CP간 세션은 지원하지 않는 유형 2.1 노드. 이것은 부속 영역 네트워크에서의 경계 노드에 접속된 주변 노드, APPN 네트워크의 APPN 네트워크 노드에 접속된 끝 노드 또는 또다른 LEN 노드나 APPN 끝 노드에 직접 접속된 피어 연결된 노드일 수 있습니다.

하위 뷰. OS/390용 DB2 UDB에서, 또 다른 뷰가 직접 또는 간접적으로 정의된 뷰.

하위 테이블. 종속 테이블에 종속되어 있거나 다른 테이블에 종속된 테이블.

하위 행. 종속 행에 종속된 행 또는 다른 행에 종속된 행.

하이퍼 공간. OS/390 환경에서, 프로그램이 버퍼로 사용할 수 있는 2 GB까지의 연속적 가상 저장영역 주소 범위. 데이터 공간처럼, 하이퍼 공간은 사용자 데이터만 보유할 수 있고, 공통 영역이나 시스템 데이터는 포함하지 않습니다. 주소 공간이나 데이터 공간과는 달리, 하이퍼 공간의 데이터는 직접 주소 지정할 수 없습니다. 하이퍼 공간에서 데이터를 조작하려면, 데이터를 4 KB 블록 단위로 주소 공간에 가져오면 됩니다.

한계 트리거. 성능 변수의 값이 사용자 정의 임계값을 초과하거나 아래에 있을 경우에 발생하는 이벤트. 임계값 트리거의 결과로 발생할 수 있는 조치는 다음과 같습니다.

- 경보 로그 파일에 정보를 기록할 수 있습니다.
- 경보 로그 창에 정보를 표시할 수 있습니다.

- 오디오 알람을 생성합니다.
- 메시지 창을 발행합니다.
- 사전에 정의된 명령이나 프로그램을 호출합니다.

할당된 커서. OS/390용 DB2 UDB에서, SQL문 ALLOCATE CURSOR를 사용하여 저장 프로시듀어 결과 세트에 대해 정의된 커서.

함수. (1) 단일 값(결과)에 대해 0개 이상의 입력 값(인수)을 사용하여 호출될 수 있는, 프로그램(함수 본문)으로 포함되는 맵핑. (2) OS/390용 DB2 UDB에서, 컬럼 함수나 스칼라 함수와 같은 특정 목적의 엔터티나 그에 해당되는 특징적 조치. 함수는 사용자가 정의하거나, 내장되어 있거나, OS/390용 DB2 UDB에서 생성됩니다.

함수 경로. 규정되지 않은 함수 호출에 대한 검색 영역을 제한하고 함수 선택 프로세스에 대한 최종 중개자를 제공하는 스키마 이름의 순서화된 목록.

함수 경로 계열. 사용자 함수 경로에서 식별되는 (또는 기본적으로 사용되는) 모든 스키마에 제공된 이름의 모든 함수.

함수 계열. 동일한 함수 이름을 가진 함수 집합. 컨텍스트는 현재 함수 경로 내에서 동일한 이름을 가진 관련된 모든 함수, 특정 스키마 내에서의 함수 세트를 사용법이 참조하는지를 판별합니다.

함수 구현자. OS/390용 DB2 UDB에서, 함수 프로그램 또는 함수 패키지의 소유자에 대한 권한 부여 ID.

함수 내용. 함수를 구현하는 코드 부분.

함수 선택. *함수 분석* 참조.

함수 시그니처. 모든 매개변수의 데이터 유형과 함께 완전한 함수 이름의 논리적 병합. 스키마 내의 각 함수에는 고유한 시그니처가 있어야 합니다.

함수 운반. 적용가능 데이터가 들어 있는 특정 노드에 대한 요청의 부속 선택 운반.

함수 정의자. OS/390용 DB2 UDB에서, CREATE FUNCTION문에 지정된 함수의 스키마 소유자에 대한 권한 부여 ID.

함수 차수. 특정 함수 인스턴스가 호출에 대해 선택되는 DBMS에 내부적인 프로세스. 함수 이름, 인수의 데이터 유형, 함수 경로 등을 사용하여 선택을 합니다. 동의어 - *함수 선택*.

함수 템플리트. 엽합 데이터베이스에서, 실행 코드가 없는 부분 함수. 사용자는 이를 데이터 소스 함수에 맵핑하여 연합 서버로부터 데이터 소스 함수가 호출될 수 있게 합니다.

함수 패키지. OS/390용 DB2 UDB에서, 함수 프로그램에 대해 DBRM 바인딩에서 초래되는 패키지.

함수 패키지 소유자. OS/390용 DB2 UDB에서, 함수 프로그램의 DBRM을 함수 패키지에 바인드하는 사용자의 권한 부여 ID.

함수 호출. 함수 내용으로 전달되는 인수 값과 함께 함수를 사용하는 것. 이 함수는 이름으로 호출됩니다.

해쉬 파티션. 행이 할당된 데이터베이스 파티션를 결정하기 위해 해쉬 함수가 파티션 키 값에 적용되는 파티션 전략.

핵심부. OS/390 환경에서, 다른 엔터티와 안전하게 통신할 수 있는 엔터티. DCE에서, 핵심부는 DCE 레지스트리 데이터베이스의 항목으로 표시되고 사용자, 서버, 컴퓨터 및 기타 사항을 포함합니다.

핵심부 이름. OS/390 환경에서, 핵심부가 DCE 보안 서비스에 알려지는 이름.

핸들. (1) 소프트웨어 시스템 내에서의 내부 구조를 나타내는 변수. (2) 테이블에서, 이미지, 오디오 또는 비디오 오브젝트를 표시하기 위해 사용되는 Extender에 의해 작성되는 문자열. 핸들은 사용자 테이블과 관리 지원 테이블에서 오브젝트마다 저장됩니다. 이 방법에서, Extender는 관리 지원 테이블에 저장되는 오브젝트에 대한 정보를 사용하여 사용자 테이블에 저장되는 핸들에 링크할 수 있습니다. (3) 텍스트 문서를 식별하는 2진값. 핸들은 Text Extender에서 컬럼이 *사용 가능*하게 될 때 텍스트 컬럼에서 각 텍스트 문서마다 작성됩니다.

행. 연속된 값들로 구성되는 테이블의 수평 구성요소. 테이블 컬럼당 하나가 지정됩니다.

행 복제. DB2 복제에서, 트랜잭션 의미 없이 Microsoft Jet용 DataPropagator에서 유지보수되는 update-anywhere 복제의 유형.

행 복제 충돌 검출. DB2 복제에서, 트랜잭션 단위가 아닌, 행 단위로, DB2 Replica에 대해 수행되는 것처럼 수행되는 충돌 검출.

행 식별자(ROWID). OS/390용 DB2 UDB에서, 행을 고유하게 식별하는 값. 이 값은 행과 함께 저장되고 변경되지 않습니다.

행 잠금. OS/390용 DB2 UDB에서, 단일 데이터 행에 대한 잠금.

행 트리거. OS/390용 DB2 UDB에서, 트리거 수준 FOR EACH ROW로 정의된 트리거.

현재 데이터. OS/390용 DB2 UDB에서, 기본 테이블 내의 데이터에 대해 현재로 식별되는 호스트 구조 내의 데이터.

현재 상태 재설정. OS/390용 DB2 UDB에서, 서브시스템 상태가 로그의 정보로부터 재구성되는 재시작 프로세스의 두 번째 단계.

현재 작업 디렉토리. 모든 상대 경로 이름이 분석되는 프로세스의 기본 디렉토리.

현재 함수 경로. 함수 및 데이터 유형에 대한 규정되지 않은 참조의 해결에 사용되는 스키마 이름의 순서화된 목록. 동적 SQL에서, 현재 함수 경로는 CURRENT FUNCTION PATH 특수 레지스터에 있습니다. 정적 SQL에서, PREP 및 BIND 명령에 대한 FUNCPATH 옵션에 정의되어 있습니다.

호스트. TCP/IP에서, 최소한 하나의 인터넷 주소가 연관되어 있는 시스템.

호스트 구조. 응용프로그램에서, Embedded SQL 명령문에 의해 참조되는 구조.

호스트 노드. SNA에서, 시스템 서비스 제어점(SSCP)을 가지고 있는 부속 영역 노드(예: MVS 및 VTAM이 있는 IBM System/390® 컴퓨터).

호스트 변수. 응용프로그램 호스트 프로그램에서, Embedded SQL 명령문에 의해 참조되는 변수. 호스트 변수는 응용프로그램에서의 프로그래밍 변수로서, 데이터베이스의 테이블과 응용프로그램 작업 영역 사이에 데이터를 전송하기 위한 기본 기제입니다.

호스트 식별자. 호스트 프로그램에서 선언된 이름.

호스트 언어. SQL문을 내포할 수 있는 프로그래밍 언어.

호스트 컴퓨터. (1) 컴퓨터 네트워크에서, 계산, 데이터베이스 액세스 및 네트워크 제어 함수와 같은 서비스를 제공하는 컴퓨터. (2) 다중 컴퓨터 설치에서 1차 컴퓨터 또는 제어 컴퓨터.

호스트 프로그램. Embedded SQL문이 들어 있는 호스트 언어로 작성된 프로그램.

호출 접속 기능(CAF). TSO 또는 MVS™ 일괄처리를 수행하는 응용프로그램에 대한 OS/390용 DB2 UDB 접속 기능. CAF는 DSN 명령 프로세서에 대한 대안으로 실행 환경에서 더 나은 제어를 제공합니다.

혼합 데이터 문자열. *혼합 문자열* 참조.

혼합 문자열. 1바이트 및 다중 바이트 문자가 혼합된 문자열. *혼합 데이터 문자열*이라고도 합니다.

홈 주소 공간. OS/390 환경에서, OS/390이 현재 *디스패치된* 것으로 인식하고 있는 저장영역.

홉(hop). APPN에서, 중간 노드가 없는 라우트의 일부. 홉은 인접 노드를 연결하는 단일 전송 그룹으로 구성됩니다.

확약. 잠금을 해제하여 작업 단위를 종료하는 조작. 이로써 그 작업 단위에 의한 데이터베이스 변경사항은 다른 프로세스에 의해 인식될 수 있습니다. 이 조작은 데이터 변경사항을 영구적으로 만듭니다.

확약 제어. Net.Data가 실행 중인 프로세스 내에서의 경계 설정. 자원 조작들은 작업 단위(UOW)의 부분입니다.

확약된 단계. OS/390용 DB2 UDB에서, 논리 작업 단위(UOW)의 결과를 확약하도록 모든 구성원에게 요청하는 다중 사이트 갱신 프로세스의 두 번째 단계.

확약점. 데이터가 일치하는 것으로 간주되는 시점. 동의어 - *일관성 지점*.

확장 복구 기능(XRF). OS/390 환경에서, 고가용성 응용프로그램 및 지시된 터미널 사이의 세션 동안 MVS, VTAM, 호스트 프로세서 또는 고가용성 응용프로그램에서의 실패 영향을 최소화하는 기능. 이 기능은 실패한 서브시스템에서 세션을 인계할 대체 서브시스템을 제공합니다.

확장 프로그램간 통신(APPC). 제품에서의 LU 6.2 아키텍처 및 다양한 구현이 특징인 일반 기능.

확장된 UNIX 코드(EUC). 길이가 1바이트에서 4바이트에 이르는 문자 세트를 지원할 수 있는 프로토콜. EUC는 실제로 코드 페이지 코드화 체계 자체가 아닌 코드 페이지 콜렉션을 지정하는 방법입니다. 이것은 PC 2바이트(DBCS) 코드 페이지 코드화 체계에 대한 UNIX 대체입니다.

환경 프로파일. 환경 변수에 대한 설정을 포함하는, Text Extender에서 제공되는 스크립트.

환경 핸들. 데이터베이스 액세스에 대한 글로벌 컨텍스트를 식별하는 핸들. 환경의 모든 오브젝트와 연관된 모든 데이터는 이 핸들과 연관됩니다.

숫자

1단계 확약중 실패. 복구 단위 상태. 복구 단위가 확약 프로세스의 1단계를 완료하기 전에 OS/390용 DB2 UDB가 실패할 경우, 재시작 시 복구 단위의 갱신만 백아웃합니다. 이러한 복구 단위를 *1단계 확약중 실패*라고 합니다.

1바이트 문자 세트(SBCS). 각 문자가 1바이트 코드로 표시되는 문자 세트.

1차 권한 부여ID. OS/390용 DB2 UDB에 대해 응용프로그램 프로세스를 식별하기 위해 사용되는 권한 부여 ID.

1차 그룹 버퍼 풀. 양방향 그룹 버퍼 풀의 경우, 캐쉬된 데이터의 결합을 유지보수하기 위해 사용되는 OS/390용 DB2 UDB 구조. 이 구조는 페이지 등록과 상호 무효화에 사용됩니다. OS/390에서는 *이전* 구조에 해당됩니다. 비교 - *2차 그룹 버퍼 풀*.

1차 로그. 데이터베이스에 대한 변경사항을 기록하는 데 사용되는 하나 이상의 로그 파일 세트. 이들 파일의 저장영역은 미리 할당되어 있습니다. 반의어 - *2차 로그*.

1차 색인. OS/390용 DB2 UDB에서, 기본 키의 고유성을 강요하는 색인.

2단계 확약. 복구가능 자원과 외부 서브시스템이 확약되는 두 단계 프로세스. 첫번째 단계 중에, 데이터베이스 관리 프로그램이 폴되어 확약할 준비가 되어야 합니다. 모든 서브시스템이 긍정적으로 응답하는 경우 데이터베이스 관리 프로그램은 확약하도록 지시합니다.

2단계 확약중 이상 실패. 복구 단위 상태. OS/390용 DB2 UDB가 1단계 확약 처리를 완료하고 2단계를 시작하기 전에 실패할 경우, 확약 조정자만 개별적 복구 단위가 확액되는지, 아니면 구간 복원되는 지를 압니다. 긴급 재시작시, OS/390용 DB2 UDB에 이러한 결정하는 데 필요한 정보가 부족할 경우, 복구 단위 상태는 OS/390용 DB2 UDB가 조정자로부터 이 정보를 확보할 때까지 *2단계 확약중 이상 실패*입니다. 재시작 시에는 여러 복구가 2단계 확약중 이상 실패할 수 있습니다.

2단계 확약중 이상 실패 분석. OS/390용 DB2 UDB에서, 2단계 확약중 이상 실패 논리 작업 단위(UOW)의 상태를 확약 상태나 구간 복원 상태로 분석하는 프로세스.

2단계 확약중 이상 실패 트랜잭션. 2단계 확약 중 한 단계가 성공적으로 완료되지만 후속 단계가 완료되기 전에 시스템이 실패하는 트랜잭션.

2단계 확약중 정상 실패. 복구 단위 상태. 2단계 확약 처리를 시작한 후 OS/390용 DB2 UDB가 실패하면, 재시작될 때 데이터에 대한 변경사항이 지속되는 것을『인식』합니다.

2바이트 문자 대형 오브젝트(DBCLOB). 2바이트 문자 연속으로서, 여기서 최고 *크기*는 2기가바이트입니다. 대형 2바이트 텍스트 오브젝트를 저장하는 데 사용할 수 있는 데이터 유형. 이를 2바이트 문자 대형 오브젝트 문자열이라고도 합니다. 이러한 문자열에는 항상 연관된 코드 페이지가 있습니다.

2바이트 문자 세트(DBCS). 각 문자가 2바이트로 표현되는 문자 집합.

2진 대형 오브젝트(BLOB). *크기* 범위가 0 바이트 - 2 기가바이트인 일련의 바이트. 이 문자열에는 연관된 코드 페이지나 문자 세트가 없습니다. 이미지, 오디오, 비디오 오브젝트가 BLOB에 저장됩니다. 비교 - *문자 대형 오브젝트(CLOB).*

2진 문자열. OS/390용 DB2 UDB에서, CCSID와 연관되지 않은 바이트 순서. 예를 들어, BLOB 데이터 유형은 2진 문자열임.

2진 정수. 추가로 작은 정수 또는 큰 정수로 분류할 수 있는 기본 데이터 유형.

2차 권한 부여 ID. OS/390용 DB2 UDB에서, 권한 부여 exit 루틴에 의해 1차 권한 부여 ID와 연관되는 권한 부여 ID.

2차 그룹 버퍼 풀. OS/390용 DB2 UDB 환경에서의 양방향 그룹 버퍼 풀의 경우, 1차 그룹 버퍼 풀에 기록되는 변경된 페이지를 백업하기 위해 사용되는 구조. 2차 그룹 버퍼 풀을 사용하는 페이지 등록이나 상호 무효화는 발생하지 않습니다. OS/390에서는 *새 구조*가 이에 해당됩니다. 비교 - *1차 그룹 버퍼 풀.*

2차 로그. 데이터베이스에 대한 변경사항을 기록하는 데 사용되는 하나 이상의 로그 파일 세트. 이들 파일의 저장영역은 1차 로그가 가득찰 때 필요에 따라 할당됩니다. 반의어 - *1차 로그.*

A

ADSM. *Tivoli Storage Manager* 참조.

Advanced Peer-to-Peer Networking(APPN). 분산된 네트워크 제어, 네트워크 자원에 대한 동적 정의, 그리고 자동화된 자원 등록 및 디렉토리 찾아보기 기능을 수행하는 SNA 확장.

Advanced Peer-to-Peer Networking(APPN) 네트워크. 상호 연결된 네트워크 노드 및 클라이언트 끝 노드 콜렉션.

APF. *권한 부여받은 프로그램 기능* 참조.

API. *API* 참조.

API(application programming interface). (1) 별도로 주문 가능한 사용권 프로그램 또는 운영 시스템에 의해 제공되는 기능 인터페이스. API는 고급 언어로 작성된 응용프로그램이 운영 체제나 사용권이 부여된 프로그램의 특정 데이터나 함수를 사용할 수 있게 합니다. (2) DB2에서, 오류 메시지 가져오기 API와 같은 인터페이스 내의 함수.

APPC. *APPC(Advanced Program-to-Program Communication)* 참조.

APPL. A VTAM® VTAM에 대해 SNA LU 6.2 프로토콜을 사용하는 응용프로그램으로 OS/390용 DB2 UDB를 정의하기 위해 사용되는 네트워크 정의 명령문.

Apply 규정자. DB2 복제에서, Apply 프로그램의 각 인스턴스에 대해 고유한 복사 작업 내역 정의를 식별하는 문자열.

Apply 프로그램. DB2 복제에서, 적용 가능한 소스에서 목표로 규칙에 따라 목표 테이블을 새로 고치거나 갱신하기 위해 사용되는 프로그램. 반의어 - *캡처 프로그램* 및 *캡처 트리거*.

APPN. *고급 피어 투 피어 네트워킹* 참조

B

BLOB. *2진 대형 오브젝트* 참조.

BSAM. *기본 순차적 액세스 메소드* 참조.

BSDS. *부트스트랩 데이터 세트* 참조.

C

CAF. *호출 접속 기능* 참조.

Capture 프로그램. DB2 복제에서, 데이터베이스 로그나 저널 레코드를 읽어서 DB2 소스 테이블 변경사항에 대한 데이터를 캡쳐하는 프로그램. 반의어 - *프로그램 적용* 및 *캡처 트리거*.

CASE 표현식. OS/390용 DB2 UDB에서, 하나 이상의 조건을 평가하여 그것을 기준으로 선택되는 다른 표현식을 허용하는 표현식.

CCD 테이블. *데이터 일관 변경 테이블* 참조.

CCSID. *코드화된 문자 세트 식별자* 참조.

CD 테이블. *변경 데이터 테이블* 참조.

CDB. *통신 데이터베이스* 참조.

CDRA. *문자 데이터 표시 아키텍처* 참조.

CEC. Central Electronic Complex. *CPC(Central Processor Complex)* 참조.

CFRM 규정. OS/390용 DB2 UDB에서, 결합 기능 구조의 할당 규칙에 관한 MVS 관리자에 의한 선언.

CHECK절. SQL에서, 테이블 점검 제한조건을 지정하는 SQL CREATE TABLE 및 SQL ALTER TABLE 명령문에 대한 확장.

CI. *제어 간격* 참조.

CICS. 중요한 비지니스 응용프로그램에 대한 관리와 온라인 트랜잭션 처리 서비스를 제공하는 IBM® 사용권 부여 프로그램. OS/390용 DB2 UDB 정보에서, 이 용어는 다음 제품을 나타냅니다.

2OS/390®용 CICS 트랜잭션 서버: OS/390용 고객 정보 제어 센터 트랜잭션 서버
CICS/ESA: 고객 정보 제어 시스템/Enterprise 시스템 아키텍처
CICS/MVS: 고객 정보 제어 시스템/다중 가상 저장영역

CICS 접속 기능. MVS 서브시스템 인터페이스(SSI)와 저장영역간 링크를 사용하여 CICS에서 OS/390용 DB2 UDB로의 요청을 처리하고 자원 확약을 조정하는 OS/390용 DB2 UDB 서브시스템.

CIDF. *제어 간격 정의 필드* 참조.

CLI. *콜 레벨 인터페이스* 참조.

CLIST. 명령 목록. TSO 타스크를 수행하기 위해 OS/390용 DB2 UDB에서 사용하는 언어.

CLOB. *문자 대형 오브젝트* 참조.

CLP. *명령행 처리기* 참조.

CLPA. *CLPA(Create Link Pack Area)* 참조.

CLPA(Create Link Pack Area). 링크 팩 페이징 가능 영역을 초기화하기 위해 IPL 동안 사용되는 옵션.

"come from" 점검. 상대 LU에서 OS/390용 DB2 UDB로의 연결에 대해 허용되는 권한 부여 ID 목록을 정의하는 LU 6.2 보안 옵션.

complete. 테이블에 관계가 있는 기본 키 값마다 한 행이 있음을 나타내는 테이블 속성. 결과적으로, 완전한 소스 테이블은 목표 테이블의 화면 정리를 새로 고치는 데 사용될 수 있습니다.

CP. *제어점* 참조.

CP 이름. 제어점 이름. 제어점 노드가 속하는 네트워크를 식별하는 네트워크 ID 규정자로 구성되는 제어점의 네트워크 규정 이름.

CPC. *CPC(Central Processor Complex)* 참조.

CPC(Central Processor Complex). 주 저장영역, 하나 이상의 중앙 프로세서, 타이머 및 채널로 구성되는 ES/3090과 같은 물리적 하드웨어 콜렉션.

CPI-C. *공통 프로그래밍 인터페이스 통신* 참조.

CPI-C 사이드 정보 프로파일. SNA에서, 원격 트랜잭션 프로그램과의 대화를 할당할 때 사용할 대화 특성을 지정하는 프로파일. 이 프로파일은 CPI 통신을 통해 통신하는 지역 트랜잭션 프로그램이 사용합니다. 상대 LU 이름(원격 LU 이름이 들어 있는 연결 프로파일 이름), 모드 이름, 원격 트랜잭션 프로그램 이름을 지정합니다.

CRC. *명령 승인 문자* 참조.

CRCR. OS/390용 DB2 UDB에서, 조건부 재시작 제어 레코드. *조건부 재시작* 참조.

CS. *커서 안정성(CS)* 참조.

CSA. *공통 서비스 영역* 참조

CT. *커서 테이블* 참조.

D

DAD(Document Access Definition). XML 형식인 XML 콜렉션의 Xml Extender 컬럼을 사용 가능하게 하게 위해 사용되는 정의.

DARI. 데이터베이스 응용프로그램 원격 인터페이스. *저장 프로시듀어*에 대한 사용되지 않는 용어.

Data Warehouse Center. 웨어하우스 구성요소에 대해 작업할 수 있게 하는 그래픽 인터페이스와 그 뒤에 있는 소프트웨어. Data Warehouse Center를 사용하여, 웨어하우스 데이터와 웨어하우스에서 데이터를 작성하는 프로세스를 정의하고 관리할 수 있습니다.

Data Warehouse Center 관리 인터페이스. Data Warehouse Center의 관리 함수에 대한 사용자 인터페이스. 인터페이스는 Data Warehouse Center 서버나 여러 관리자를 위한 다른 많은 머신에 있을 수 있습니다.

Data Warehouse Center 등록 정보. 기술 메타데이터를 포함하는 웨어하우스 제어 데이터베이스와 같이, Data Warehouse Center으 세션들에 적용되는 속성. 또한 *등록 정보* 참조.

Data Warehouse Center 프로그램. Data Warehouse Center에서 시작될 수 있으며 자동으로 정의되는, Data Warehouse Center에서 제공되는 프로그램으로, 예를 들면, DB2 로드 프로그램과 변환기가 있습니다.

DataJoiner. 분산 데이터에 대한 클라이언트 응용프로그램 통합 액세스를 제공하며, 이질적인 환경의 단일 데이터베이스 이미지를 제공하는 별도로 제공되는 제품. DataJoiner를 사용할 때, 클라이언트 응용프로그램은 여러 데이터베이스 관리 시스템에 분산되어 있거나 데이터가 지역 데이터인 것처럼 단일 원격 데이터 소스를 갱신하는 데이터를 단일 SQL 문을 사용하여 조인할 수 있습니다.

DataJoiner 복제 관리(DJRA) 도구. 다양한 복제 관리 타스크를 수행하기 위해 사용할 수 있는 데이터베이스 관리 도구. 제어 센터와는 달리, DJRA 도구를 사용하여 IBM 데이터베이스가 아닌 다른 데이터베이스에 대한 복제를 관리할 수 있습니다. 반의어 - **제어 센터**.

DATALINK. 데이터베이스에서 데이터베이스 외부에 저장된 파일로의 논리 참조가 가능하도록 하는 DB2 데이터 유형.

DB2 CLI. DB2 콜 레벨 인터페이스. DB2 전체 기능을 이용하는 DB2 제품군에 대한 대체 SQL 인터페이스.

DB2 Connect. DRDA 응용프로그램 서버(AS)에 저장된 데이터를 읽고 갱신하기 위해 클라이언트 응용프로그램에 대해 필요한 함수를 제공하는 제품(DRDA 응용프로그램 리퀘스터(AR) 지원).

DB2 Extender. 이미지, 오디오 및 비디오 데이터와 같이, 일반적인 숫자 및 문자 데이터를 벗어나는 데이터 유형을 저장하고 검색하기 위해 사용할 수 있는 프로그램.

DB2 PM. OS/390용 DB2 UDB에서, DATABASE 2 성능 모니터링.

DB2 SDK. *DB2 응용프로그램 개발 클라이언트*에서 참조하십시오.

DB2 명령. 사용자가 OS/390용 DB2 UDB 시작 또는 중정, 현재 사용자에 대한 정보 표시, 데이터베이스 시작 또는 중지, 데이터베이스 상태에 대한 정보 등과 같은 작업을 수행할 수 있게 하는 OS/390용 DB2 UDB 서브시스템에 대한 지시사항.

DB2 스레드. 응용프로그램의 연결을 설명하고, 그 과정을 추적하며, 자원 함수를 처리하고, 액세스 가능성을 OS/390용 DB2 UDB 자원 및 서비스로 구분하는 OS/390용 DB2 UDB 구조.

DB2 응용프로그램 개발 클라이언트 (DB2 SDK). 개발자가 데이터베이스 응용프로그램을 작성하는 것을 돕는 도구의 콜렉션.

DB2I. OS/390용 DB2 UDB에서, DATABASE 2 대화식.

DB2I Kanji 기능. OS/390용 DB2 UDB에서, 패널과 사이트에 Kanji로 DB2I 패널을 표시할 수 있게 하는 작업이 있는 테이프.

DB2UEXIT. 아카이브 로그 파일의 이동 또는 검색을 위한 데이터베이스 관리 프로그램 호출을 하는 선택적인 사용자 작성 실행 가능 프로그램.

DB2간 R/W 관계. OS/390용 DB2 UDB에서, 데이터 공유 그룹의 여러 구성원이 열고, 그 구성원들 중 최소한 하나의 구성원이 기록하기 위해 연 테이블 공간, 색인 또는 파티션에 있는 데이터의 등록 정보.

DBA. *데이터베이스 관리자* 참조.

DBA 유틸리티. DB2 사용자가 데이터베이스 및 데이터베이스 관리 프로그램 인스턴스를 구성하고, 지역 및 원격 데이터베이스에 액세스하는 데 필요한 디렉토리를 관리하며, 데이터베이스나 테이블 공간의 백업 및 복구, 그래픽 인터페이스를 사용한 시스템상의 미디어 관리 등을 할 수 있도록 하는 도구. 이 도구가 제공하는 타스크들은 제어 센터로부터 액세스할 수 있습니다.

DBCLOB. *2바이트 문자 대형 오브젝트* 참조.

DBCS. *2바이트 문자 세트* 참조.

DBD. *데이터베이스 설명자* 참조.

DBID. 데이터베이스 식별자.

DBMS. 데이터베이스 관리 시스템. *데이터베이스 관리 프로그램* 참조.

DBMS 인스턴스 연결. 응용프로그램 및 DB2 인스턴스가 소유하는 에이전트 프로세스 또는 스레드 사이의 논리적 연결.

DBRM. *데이터베이스 요청 모듈* 참조.

DCE. *분산 컴퓨팅 환경(DCE)* 참조.

DCE 티켓. OS/390 환경에서, 초기화하는 핵심부의 정체를 전송하는 투명한 응용프로그램 기제. 단순 티켓에는 핵심부의 정체, 세션 키, 시간소인 및 기타 정보가 있으며, 이 티켓은 목표의 비밀 키를 사용하여 증명됩니다.

DCE(Distributed Computing Environment). 이질적인 전산 환경에서 분산 응용프로그램의 작성, 사용 및 유지보수를 지원하는 도구 및 서비스 집합. DCE는 운영 체제 및 네트워크와 독립적이며, 이질적인 플랫폼을 통해 상호 조작성 및 이식성을 제공합니다.

DCLGEN. *선언 생성기* 참조.

DDF. *분산 데이터 기능* 참조.

DDL. *데이터 정의 언어* 참조.

DDL. 데이터베이스 내에서의 데이터 및 데이터간 관계를 서술하기 위한 언어. 동의어 - *데이터 설명 언어*.

ddname. *데이터 정의 이름* 참조.

DFHSM. OS/390 환경에서, 데이터 기능 계층적 저장영역 관리 프로그램.

DFP. OS/390 환경에서, 데이터 기능 제품.

DJRA 도구. 다양한 복제 관리 타스크를 수행하기 위해 사용할 수 있는 데이터베이스 관리 도구. 제어 센터와는 달리, DJRA 도구를 사용하여 IBM 데이터베이스가 아닌 다른 데이터베이스에 대한 복제를 관리할 수도 있습니다. 반의어 - **제어 센터**.

DLC. *데이터 링크 제어* 참조.

DLU. *종속 논리 단위* 참조.

DML. *DML* 참조.

DML. 데이터 조작에 사용되는 SQL문의 부속 집합.

DMS 테이블 공간. *데이터베이스 관리 공간 테이블 공간* 참조.

DNS. *도메인 이름 시스템* 참조.

Domino™ *Go* 웹Server. Lotus® Corp. 및 IBM에서 제공되는 웹 서버로, 일반 연결과 보안 연결 둘 다를 제공합니다. ICAPI 및 GWAPI는 이 서버내에서 제공되는 인터페이스입니다.

DRDA. *Distributed Relational Database Architecture*에서 참조하십시오.

DRDA 액세스. OS/390용 DB2 UDB에서, SQL문으로 다른 위치에 연결하여 이전에 그 위치에 바인드된 패키지를 실행할 수 있는 분산 데이터 액세스 방법. SQL CONNECT 또는 세 부분 이름 명령문은 응용프로그램 서버(AS)를 식별하기 위해 사용되고, SQL문은 이전에 해당 서버에서 바인드된 패키지를 사용하여 실행됩니다. 반의어 - *개인 프로토콜 액세스*.

DRDA(Distributed Relational Database Architecture). 원격 데이터에 대한 투명한 액세스를 제공하기 위해 형식과 프로토콜을 정의하는 아키텍처. DRDA는 응용프로그램 리퀘스터 기능과 응용프로그램 서버(AS) 기능이라는 두 가지 기능을 정의합니다.

DSN. (1) OS/390용 DB2 UDB에 대한 기본값 서브시스템 이름. (2) OS/390용 DB2 UDB의 TSO 명령 이름. (3) OS/390용 DB2 UDB 모듈 및 매크로 이름의 처음 세 자.

DUOW. *분산 작업 단위* 참조.

E

EA 사용 테이블 공간. OS/390용 DB2 UDB에서, 확장 주소 지정 가능성에 사용하고 4 GB 이상인 개별적 파티션(LOB 테이블 공간의 경우 조각)을 포함하는 테이블 공간이나 색인 공간.

EBCDIC. 확장 2진 코드 10진 상호변경 코드. 256개의 8비트 문자로 된 코드화 문자 세트.

EDM 풀. OS/390용 DB2 UDB에서, 데이터베이스 설명자, 응용프로그램 플랜, 권한 부여 캐쉬, 응용프로그램 패키지 및 동적 명령문 캐쉬에 사용되는 주 저장영역 풀.

EID. 이벤트 식별자.

Embedded SQL. 응용프로그램 내에서 코드화된 SQL문. *정적 SQL* 참조.

EN. *끝 노드* 참조.

enclave. 언어 환경(OS/390용 DB2 UDB에서 사용되는)에서, 독립적인 루틴 콜렉션으로, 기본 루틴으로 지시된 콜렉션입니다. enclave는 프로그램 또는 수행 단위와 유사합니다.

EOM. 메모리의 끝.

EOT. 타스크의 끝.

equijoin. 술어에 등호 연산자가 있는 조인(예: T1.C1 = T2.C2).

escape 문자. SQL 분리 식별자를 묶기 위해 사용되는 기호. Escape 문자는, 사용자가 큰 따옴표나 작은 따옴표 중 하나인 기호를 지정하는 COBOL 응용프로그램을 제외하고, 큰 따옴표입니다.

ESDS. OS/390 환경에서, 항목 순서화 데이터 세트.

ESMT. OS/390 환경에서, IMS의 외부 서브시스템 모듈 테이블.

EUC. *Extended UNIX® 코드* 참조.

exit 루틴. 다른 프로그램(예: OS/390용 DB2 UDB)에서 제어를 받아서 특정 함수를 수행하는 프로그램.

Explain 스냅샷. SQL 조회 및 관련 정보의 현재 내부 표현 캡쳐. 이 정보는 Visual Explain 도구에 필수적입니다.

Explain된 통계. 명령문이 Explain될 때 SQL문에서 참조되었던 데이터베이스 오브젝트에 대한 통계.

extent. 테이블 공간의 컨테이너 내에서 단일 데이터베이스 오브젝트로의 공간 할당. 이 할당은 여러 페이지로 구성됩니다.

extent 맵. 테이블 공간의 각 오브젝트에 extent 할당을 기록하는 테이블 공간 내에 저장된 메타데이터 구조.

F

fallback. 현재 릴리스로의 이주를 시도하거나 완료한 후 OS/390용 DB2 UDB의 이전 릴리스로 리턴하는 프로세스.

fullselect. 집합 연산자에 의해 결합되는 부속 선택, 값 절 또는 이 둘의 조합.

G

GBP. 그룹 버퍼 풀.

GBP 종속. OS/390용 DB2 UDB에서, 그룹 버퍼 풀에 종속되는 페이지 세트나 페이지 세트 파티션의 상태. 읽기/쓰기 관계가 이 페이지 세트에 대한 DB2 서브시스템 사이에 활동중이거나, 페이지 세트에 아직 DASD 밖에서 유형변환되지 않은 그룹 버퍼 풀 내의 변경된 페이지가 있습니다.

Getpage. OS/390용 DB2 UDB가 데이터 페이지에 액세스하는 조작.

GIMSMP. OS/390 환경에서, 프로그래밍 시스템에 대한 변경사항을 설치, 변경 및 제어하기 위한 기본 도구인 System Modification Program/Extended에 대한 로드 모듈 이름.

GTF. *GTF(generalized trace facility)* 참조.

GTF(generalized trace facility). OS/390 에서, I/O 인터럽트, SVC 인터럽트, 프로그램 인터럽트 또는 외부 인터럽트와 같은 중요한 시스템 이벤트를 기록하는 서비스 프로그램.

GWAPI. Domino *Go* Web Server API.

H

HSM. OS/390 환경에서, 계층적 저장영역 관리 프로그램.

I

ICAPI. 인터넷 연결 API.

ICF. OS/390 환경에서, 통합된 카탈로그 기능.

IDCAMS. OS/390 환경에서, 액세스 메소드 서비스 명령을 처리하기 위해 사용되는 IBM 프로그램. 이것은 TSO 터미널이나 사용자의 응용프로그램에서 작업 또는 작업 단계로 호출될 수 있습니다.

IDCAMS LISTCAT. OS/390 환경에서, 액세스 메소드 서비스 카탈로그에 포함된 정보를 확보하기 위한 기능.

IFCID. OS/390용 DB2 UDB에서, IFCID(instrumentation facility component identifier).

IFCID(instrumentation facility component identifier). OS/390용 DB2 UDB에서, 추적할 수 있는 이벤트의 추적 레코드를 명명하고 식별하는 값. START TRACE 및 MODIFY TRACE 명령의 매개변수처럼, 해당되는 이벤트가 추적됨을 지정합니다.

IFI. OS/390용 DB2 UDB에서, IFI(instrumentation facility interface).

IFI 호출. OS/390용 DB2 UDB에서, 정의된 함수 중 하나에 의한 IFI(instrumentation facility interface)의 호출.

IFI(instrumentation facility interface). 프로그램에서 OS/390용 DB2 UDB에 대한 온라인 추적 데이터를 확보하고, OS/390용 DB2 UDB 명령을 제출하며, OS/390용 DB2 UDB에 데이터를 전달할 수 있게 하는 프로그래밍 인터페이스.

IFP. OS/390 환경에서, IMS 빠른 경로.

ILU. *독립 논리 장치* 참조.

IMS. 정보 관리 시스템.

IMS DB. 정보 관리 시스템 데이터베이스.

IMS TM. 정보 관리 시스템 트랜잭션 관리 프로그램.

IMS 접속 기능. OS/390 서브시스템 인터페이스(SSI) 프로토콜과 상호 메모리 링크를 사용하여 IMS에서 OS/390용 DB2 UDB로의 요청을 처리하고 자원 확약을 조정하는 OS/390용 DB2 UDB 부속 구성요소.

Information Catalog Manager. 비지니스 정보를 구성, 유지보수, 찾기 및 사용하기 위한 응용프로그램.

IP. *인터넷 프로토콜* 참조.

IP 주소. TCP/IP 호스트를 고유하게 식별하는 4 바이트 값.

IPX. 인터네트워크 패킷 교환.

IPX(Internetwork Packet Exchange). 데이터를 원격 노드에 전달하기 위해 NetWare LAN 환경에서 사용되는 연결 없는 데이터그램 프로토콜. IPX는 데이터 패킷을 보내기 위해 최선을 노력을 하지만, 데이터의 신뢰성 있는 전달은 보장하지 못합니다.

IRLM. OS/390용 DB2 UDB에서, 내부 자원 잠금 관리 프로그램.

ISAPI. Microsoft® 인터넷 서버 API.

ISPF. OS/390 환경에서, 대화식 시스템 생산성 기능.

ISPF(Interactive System Productivity Facility). OS/390 환경에서, 대화식 대화 상자 서비스를 제공하는 IBM 사용권 부여 프로그램.

ISPF/PDF. OS/390 환경에서, 대화식 시스템 생산성 기능/프로그램 개발 기능.

I/O 병렬 처리. *병렬 I/O*에서 참조하십시오.

J

JCL. *작업 제어 언어* 참조.

JES. *작업 항목 서브시스템* 참조.

K

KSDS. *키순 데이터 세트*.

L

LCID. OS/390 환경에서, 로그 제어 간격 정의.

LDS. *선형 데이터 세트* 참조.

LEN 노드. *하위 네트워크(LEN)* 노드 참조.

Live Connection. 연결 관리 프로그램과 여러 클리에트로 구성되는 Net.Data 구성요소. Live Connection은 데이터베이스와 Java® 가상 기계 연결의 재사용을 관리합니다.

LOB. *대형 오브젝트(LOB)* 참조.

LOB 위치 지정자. 응용프로그램이 데이터베이스 시스템에서 대형 오브젝트(LOB) 값을 조작할 수 있도록 허용하는 메카니즘. LOB 위치 지정자는 단일 LOB 값을 나타내는 간단한 토큰 값입니다. 응용프로그램은 LOB 위치 지정자를 호스트 변수 내로 검색하므로 위치 지정자를 사용하여 SQL 함수를 연관되는 LOB 값에 적용할 수 있습니다.

LOB 잠금. OS/390용 DB2 UDB에서, LOB 값에 대한 잠금.

LOB 테이블 공간. OS/390용 DB2 UDB에서, 관련된 기본 테이블의 특정 LOB 컬럼에 대한 모든 데이터를 포함하는 테이블 공간.

long 문자열. (1) 최대 길이가 254바이트 이상인 가변 길이 문자열. (2) OS/390용 DB2 UDB에서, 실제 길이가 255 바이트나 127 2바이트보다 큰 문자열이나, 최대 길이가 255 바이트나 127 2바이트보다 큰 문자열. LOB 컬럼, LOB 호스트 변수 또는 LOB에 대해 평가하는 표현식은 LONG 문자열로 간주됩니다.

LPL. *논리 페이지 목록* 참조.

LRECP. *논리 복구 보류* 참조.

LRH. OS/390용 DB2 UDB에서, 로그 레코드 헤더.

LRSN. *로그 레코드 순차 번호* 참조.

LU. *논리 단위* 참조.

LU 6.2. *논리 단위 6.2* 참조.

LU 유형. 특히, 주어진 세션에 대해 지원하는 특정 SNA 프로토콜 및 옵션의 관점에서 논리 장치를 분류한 것.

- 세션 활성화 요청에 허용되는 값
- 데이터 스트림 제어, 함수 관리 헤더, 요청 단위 매개변수, 센스 데이터 값의 사용.
- 함수 관리 헤더와 연결된 프로토콜과 같은 프리젠테이션 서비스 프로토콜

LU 이름. OS/390 환경에서, VTAM이 네트워크에서 노드를 참조하는 이름. 반의어 - *위치 이름*.

LUW. *논리 작업 단위* 참조.

LUWID. *논리 작업 단위 식별자* 참조.

L-잠금. *논리 잠금* 참조.

M

MBCS. *복수 바이트 문자 세트* 참조.

MODEENT. OS/390 환경에서, 로그온 모드 이름을 세션 프로토콜을 나타내는 매개변수 세트와 연관시키는 VTAM 매크로 명령어. MODEENT 매크로 명령어 세트는 로그온 모드 테이블을 정의합니다.

MPP. (1) 대량의 병렬 처리. (2) IMS내의 OS/390 환경에서, 메시지 처리 프로그램.

MSS. OS/390 환경에서, 대량 저장영역 서브시스템.

MTO. OS/390 환경에서, 마스터 터미널 운영자.

MVS. OS/390의 부분인 다중 가상 저장영역.

MVS/ESA™. OS/390의 부분인 다중 가상 저장영역/엔터프라이즈 시스템 아키텍처.

N

NAU. *네트워크 주소지정 가능 단위* 참조.

NDS. *네트워크 디렉토리 서비스* 참조.

NETID. 네트워크 식별자. *네트워크 이름* 참조.

NID. *네트워크 식별자* 참조.

NN. *네트워크 노드* 참조.

NRE. OS/390 환경에서, 네트워크 복구 요소.

NSAPI. Netscape API.

NUL. C 언어에서, 문자열의 끝을 표시하는 단일 문자.

NUL 종료 호스트 변수. OS/390용 DB2 UDB에서, 데이터의 끝이 NUL 종료자 존재로 표시되는 가변 길이 호스트 변수.

NUL 종료자. C 언어에서, 문자열의 끝을 나타내는 값. 문자열의 경우, NUL 종료자는 X'00'입니다.

NULLIF. OS/390용 DB2 UDB에서, 두 개의 제공된 표현식을 평가하여 인수가 같으면 NULL을 리턴하고 그렇지 않으면 첫번째 인수의 값을 리턴하는 스칼라 함수.

O

OASN(원점 응용프로그램 스케줄 번호). IMS가 있는 OS/390 환경에서, IMS의 마지막 콜드 시동 후에 각 IMS 스케줄마다 순차적으로 지정되는 4 바이트 숫자. OASN은 작업 단위(UOW)에 대한 식별자로 사용됩니다. 8 바이트 형식에서, 처음 4 바이트에는 스케줄 번호가 있고 마지막 4 바이트에는 현재 스케줄 동안의 IMS 동기 지점(*확약 지점*) 수가 있습니다. OASN은 IMS 연결에 대한 NID의 부분입니다.

OBID. OS/390용 DB2 UDB에서, 데이터 오브젝트 식별자.

ODBC. *ODBC* 참조.

ODBC 드라이버. ODBC 함수 호출을 구현하고 데이터 소스와 상호작용하는 드라이버.

ODBC(Open Database Connectivity). SQL 선행 처리기를 사용하지 않아도 되는 호출 가능한 SQL을 사용하여 데이터베이스 관리 시스템에 액세스할 수 있게 하는 API. ODBC 아키텍처는 런타임 시 데이터베이스 관리 시스템에 대한 사용자들의 선택사항에 응용프로그램을 링크하는 *데이터베이스 드라이버*라고 하는 모듈을 사용자가 추가할 수 있도록 합니다. 응용프로그램은 지원되는 모든 데이터베이스 관리 시스템 모듈에 직접 링크되지 않아도 됩니다.

OLAP. *온라인 분석 처리* 참조.

P

PCT. CICS에서, 프로그램 제어 테이블.

PDS. *파티션된 데이터 세트* 참조.

PLT. CICS에서, 프로그램 목록 테이블.

point-in-time 테이블. DB2 복제에서, 내용이 소스 테이블의 일부나 모두와 일치하는 목표 테이블의 유형으로, 추가되는 시스템 컬럼은 소스 시스템에서 특정 행이 삽입되거나 갱신될 때 대략의 시간을 식별합니다.

PPT. (1) CICS에서, 프로그램 처리. (2) OS/390에서, 프로그램 등록 정보 테이블.

prepare. (1) SQL 컴파일러로 제출하여, 텍스트 형식에서 실행가능 형식으로 SQL문을 변환하는 것. (2) OS/390용 DB2 UDB에서, 모든 구성원이 확약 준비를 하도록 요청되는 2단계 확약 프로세스의 첫번째 단계.

protocol.ini. 모든 프로토콜 및 매체 액세스 제어(MAC) 시스템 모듈에 대한 LAN 구성 및 바인딩 정보를 포함하는 파일.

PSRCP. OS/390용 DB2 UDB에서, 페이지 세트 복구 보류.

PU. *물리적 단위* 참조.

PU 유형. SNA에서, 상주하고 있는 노드의 유형에 따른 물리 장치의 분류.

P-잠금. *물리적 잠금* 참조.

Q

QSAM. 대기행렬에 대기된 순차적 액세스 메소드.

quiesce. 작업에 대한 새로운 요청을 거부하면서, 조작이 정상 완료되도록 하는 프로세스 종료 작업.

quiesce 구성원 상태. OS/390용 DB2 UDB에서, 데이터 공유 그룹 구성원의 상태. 활동중인 구성원은 STOP DB2 명령이 실패 없이 효과를 발휘하게 될 경우 Quiesce 상태가 됩니다. 명령 효과가 발휘되기 전에 구성원 타스크, 주소 공간 또는 OS/390 시스템이 실패할 경우, 구성원 상태는 실패가 됩니다.

R

RACF®. OS/390 환경에서, 자원 액세스 제어 기능.

RAMAC®. OS/390 환경에서, 엔터프라이즈 디스크 저장영역 시스템 제품의 IBM 계열.

RBA. *상대 바이트 주소* 참조.

RCT. CICS 접속 기능이 있는 OS/390용 DB2 UDB에서, 자원 제어 테이블.

RDB. *관계형 데이터베이스* 참조.

RDBMS. *관계형 데이터베이스 관리 시스템* 참조.

RDBNAM. *관계형 데이터베이스 이름* 참조.

RDF. OS/390용 DB2 UDB에서, 레코드 정의 필드.

RECP. OS/390용 DB2 UDB에서, 복구 보류.

redo. OS/390용 DB2 UDB에서, 데이터 무결성을 위해 변경사항이 DASD 미디어에 다시 적용됨을 나타내는 복구 단위 상태.

REORG 보류(REORP). OS/390용 DB2 UDB에서, 재구성해야 하는 오브젝트에 대한 SQL 액세스와 대부분의 유틸리티 액세스를 제한하는 상태.

REORP. *REORG 보류* 참조.

RESTP. *재시작 보류* 참조.

RID. *레코드 식별자* 참조.

RID 풀. *레코드 식별자 풀* 참조.

RLF. *자원 한계 기능* 참조.

RO. OS/390용 DB2 UDB에서, 읽기 전용 액세스.

ROWID. *행 식별자* 참조.

RR. *반복 읽기(RR)* 참조.

RRE. IMS가 있는 OS/390 환경에서, residual 복구 항목.

RRSAF. OS/390용 DB2 UDB와, OS/390 시스템에서 OS/390 RRS도 사용하는 다른 모든 자원 관리자 사이의 자원 확약을 조정하기 위해 OS/390 Transaction Management and Recoverable Resource Manager Services를 사용하는 OS/390용 DB2 UDB의 부속 구성요소인, 복구 가능한 자원 관리 프로그램 서비스 접속 기능.

RS. *읽기 안전성(RS)* 참조.

RUOW. *원격 작업 단위(RUOW)* 참조.

S

sargable. 검색 인수로서 평가될 수 있는 술어.

SBCS. *1바이트 문자 세트* 참조.

SCA. OS/390용 DB2 UDB에서, 공유된 통신 영역.

SDK. *Software Developer's Kit(SDK)* 참조.

SDWA. OS/390 환경에서, 시스템 진단 작업 영역.

shift-in 문자. 후속 바이트가 SBCS 문자를 나타냄을 표시하기 위해 EBCDIC 시스템에서 사용되는 특수 제어 문자 (X'0F'). 반의어 - *shift-out 문자*.

shift-out 문자. 다음 shift-in 제어 문자까지의 후속 바이트가 DBCS 문자를 나타냄을 표시하기 위해 EBCDIC 시스템에서 사용되는 특수 제어 문자(X'0E'). 반의어 - *shift-in 문자*.

short 문자열. (1) 최대 길이가 254 바이트보다 작거나 같은 고정 길이 문자열이나 가변 길이 문자열. (2) OS/390용 DB2 UDB에서, 실제 길이가 255 바이트나 127 2바이트이거나 이보다 작은 문자열이나, 최대 길이가 255 바이트나 127 2바이트이거나 이보다 작은 가변 길이 문자열. 길이에 관계없이, LOB 문자열은 short 문자열이 아닙니다.

SMF. OS/390 환경에서, 시스템 관리 기능.

SMS. OS/390 환경에서, 저장 관리 서브시스템.

SMS 테이블 공간. 공간이 운영 시스템에 의해 관리되는 테이블 공간. 이 저장영역 모델은 서브디렉토리 아래에 작성된 파일에 기초하며, 파일 시스템에 의해 관리됩니다. 반의어 - *데이터베이스 관리 공간(DMS) 테이블 공간*.

SMS 테이블 공간. *SMS 테이블 공간* 참조.

SNA. *SNA(Systems Network Architecture)* 참조.

SNA 네트워크. SNA(Systems Network Architecture) 프로토콜 및 형식을 따르는 사용자 응용프로그램 네트워크 부분. 사용자들 사이의 신뢰성 있는 데이터 전송을 가능하게 하고, 다양한 네트워크 구성 자원을 제어하기 위한 프로토콜을 제공합니다. SNA 네트워크는 네트워크 주소지정가능 장치(NAU), 게이트웨이 함수, 중간 세션 경로지정 함수 구성요소, 전송 네트워크 등으로 구성됩니다.

Software Developer's Kit(SDK). DB2 Connect 제품을 통해 호스트 관계형 데이터베이스를 포함하여 원격 데이터베이스 서버에 액세스할 수 있도록 클라이언트 워크스테이션 상에서 응용프로그램이 개발되도록 허용하는 응용프로그램 개발 제품.

spill 파일. DB2 복제에서, 여러 목표 테이블에 대해 데이터를 갱신하기 위한 소스로 사용되는 Apply 프로그램에서 작성되는 임시 파일.

Spreadsheet Add-in. OLAP Starter 킷에서, Microsoft Excel 및 Lotus 1-2-3과 병합되어 데이터의 다차원 분석을 허용하는 소프트웨어. 소프트웨어 라이브러리는 스프레드시트에 대한 메뉴 추가 기능으로 표시되어, 그러한 다차원 분석 기능을 연결, 확대 및 계산으로 제공합니다.

SPUFI. OS/390용 DB2 UDB에서, 파일 입력을 사용하는 SQL 프로세서.

SQL. 관계형 데이터베이스 데이터를 정의 및 조작하기 위한 표준화 언어.

SQL. *SQ* 참조.

SQL escape 문자. OS/390용 DB2 UDB에서, SQL 분리 식별자를 묶기 위해 사용되는 기호. 이 기호는 큰 따옴표(")입니다. 비교 - *escape 문자*.

SQL ID. *SQL 권한 부여 ID*.

SQL 경로. OS/390용 DB2 UDB에서, 사용자 정의 함수, 구별 유형 및 저장 프로시듀어에 대한 규정되지 않은 참조 사항의 분석에 사용되는 정렬된 스키마 이름 목록. 동적 SQL에서, 현재 경로는 CURRENT PATH 특수 레지스터에 있습니다. 정적 SQL에서, PATH 바인드 옵션에 정의되어 있습니다.

SQL 권한 부여 ID(SQL ID). OS/390용 DB2 UDB에서, 일부 상황에서 동적 SQL문을 점검하기 위해 사용되는 권한 부여 ID.

SQL 루틴. OS/390용 DB2 UDB에서, SQL로 작성되는 코드를 기초로 하는 사용자 정의 함수(UDF) 또는 저장 프로시듀어.

SQL 리턴 코드. SQLCODE 또는 SQLSTATE.

SQL 문자열 분리문자. OS/390용 DB2 UDB에서, SQL 문자열 상수를 묶기 위해 사용되는 기호. SQL 문자열 분리문자, 사용자가 큰 따옴표나 작은 따옴표 중 하나인 기호를 지정하는 COBOL 응용프로그램을 제외하고, 작은 따옴표(')입니다.

SQL 설명자 영역(SQLDA). (1) 일부 SQL문을 처리하는 데 사용되는 변수 집합. SQLDA는 동적 SQL 프로그램용입니다. (2) 입력 변수, 출력 변수 또는 결과 테이블의 컬럼을 설명하는 구조.

SQL 연결. OS/390용 DB2 UDB에서, 응용프로그램 프로세스와 지역 또는 원격 응용프로그램 서버(AS) 사이의 연관.

SQL 처리 대화. 응용프로그램을 통하거나 동적 조회 요청에 의한, OS/390용 DB2 UDB 데이터의 액세스를 요구하는 대화.

SQL 통신 영역(SQLCA). 응용프로그램에 데이터베이스 관리 프로그램으로부터의 요청이나 SQL문 실행에 대한 정보를 제공하는 변수 집합.

SQLCA. *SQL 통신 영역*.

SQLDA. *SQL 설명자 영역* 참조.

SSCP. *시스템 서비스 제어점* 참조.

SSI. OS/390 환경에서, 서브시스템 인터페이스.

SSM. OS/390용 DB2 UDB에서, 서브시스템 구성원.

Stored Procedure Builder. 저장 프로시듀어 작성, 지역 및 원격 DB2 서버에서 저장 프로시듀어 빌드, 기존 저장 프로시듀어 수정 및 재빌드, 그래픽 인터페이스를 사용하여 설치된 저장 프로세듀어의 실행 테스트 및 디버깅 등의 작업을 수행하는 도구. 이 도구는 독립형으로, 다양한 통합 개발 환경에서 액세스될 수 있습니다.

Stored Procedure Builder 프로젝트. 연결 정보와 데이터베이스에서 성공하지 못한 저장 프로시듀어 오브젝트를 포함하는 파일로, Stored Procedure Builder에서 작성됩니다.

SYS1.DUMPxx 데이터 세트. OS/390 환경에서, 시스템 덤프를 포함하는 데이터 세트.

SYS1.LOGREC. OS/390 환경에서, 프로그램과 하드웨어 오류에 대한 중요한 정보를 포함하는 서비스 보조 기능.

Sysplex. *병렬 Sysplex* 참조.

Sysplex 조회 병렬 처리. 여러 개의 OS/390용 DB2 UDB 서브시스템에서 여러 타스크를 사용하여 수행되는 단일 조회의 병렬 실행. 또한 *조회 CP 병렬 처리* 참조.

T

TCB. *타스크 제어 블록* 참조.

TCP/IP. *전송 제어 프로토콜/인터넷 프로토콜*.

TCP/IP 포트. TCP/IP 호스트 내에서 일반 사용자나 TCP/IP 네트워크 응용프로그램을 식별하는 2 바이트 값.

timeron. 데이터베이스 서버가 동일한 조회에 대해 두 계획을 실행하기 위해 필요한 자원 또는 비용의 상대 측정치를 제공하는 데 사용되는 측정 단위. 예측에서 계산되는 자원에는 가중치가 부여된 프로세서와 입출력 비용이 포함됩니다.

Tivoli Storage Manager(TSM). 이질적인 환경에서 저장영역 관리 및 데이터 액세스 서비스를 제공하는 클라이언트/서버 제품. TSM은 다양한 통신 메소드를 지원하고, 파일의 백업 및 저장영역을 관리하기 위한 관리 기능을 제공하며, 백업 조작을 스케줄링하기 위한 기능을 제공합니다.

TM 데이터베이스. *트랜잭션 관리 프로그램 데이터베이스* 참조.

TMP. OS/390 환경에서, 터미널 모니터 프로그램.

to-do. 복구할 수 있는 OS/390용 DB2 UDB 자원에 대해 복구 단위별로 수행된 변경사항이 2단계 확약중 이상 실패하여, 확약 조정자에 의해 판별되는 대로 DASD 미디어에 적용되어야 하거나 백아웃되어야 함을 나타내는 복구 단위 상태.

TP. *트랜잭션 프로그램* 참조.

TSO. OS/390 환경에서, 시간 공유 옵션.

TSO 접속 기능. DSN 명령과 DB2I로 구성되는 OS/390용 DB2 UDB 기능. CICS 또는 IMS 환경에 대해 작성되지 않은 응용프로그램은 TSO 접속 기능 하에 실행될 수 있습니다.

U

UDF. *사용자 정의 함수* 참조.

UDT. *사용자 정의 유형* 참조.

unit-of-work table. 데이터베이스 로그 또는 저널로부터 읽히는 확약 레코드가 들어 있는 소스 서버에서의 복제 제어 테이블. 레코드에는 unit-of-work table과 데이터 변경 테이블을 조인하여 트랜잭션-데이터 일관 변경 테이블을 산출하는 데 사용할 수 있는 복구 단위 ID가 포함됩니다. DB2의 경우, 작업 단위 테이블에는 선택적으로 감사에 유용할 수 있는 상관 ID가 포함됩니다.

upstream. OS/390용 DB2 UDB에서, 다른 복구 또는 자원 관리 프로그램 외에도, 2단계 확약 실행을 해야 하는 동기 지점 트리 내의 노드.

UR. *미확약 읽기(UR)* 참조.

URE. OS/390용 DB2 UDB에서, 복구 단위 요소.

URID(복구 단위 ID). OS/390용 DB2 UDB에서, 복구 단위에 대한 첫번째 로그 레코드의 LOGRBA. URID는 또한 해당되는 복구 단위에 대한 모든 후속 로그 레코드에 표시됩니다.

UT. OS/390용 DB2 UDB에서, 유틸리티 전용 액세스.

UTC. *세계 표준시*에서 참조하십시오.

V

value. (1) SQL에서 조작되는 가장 작은 데이터 단위. (2) 컬럼 및 행의 상호교차 지점에 있는 특정 데이터 항목.

Visual Explain. 주어진 SQL문의 액세스 플랜에 대한 상세한 정보를 표시하고 분석하기 위해 데이터베이스 관리자가 응용프로그램 프로그래머가 그래픽 인터페이스를 사용할 수 있도록 하는 도구. 이 도구가 제공하는 타스크들은 제어 센터로부터 액세스할 수 있습니다.

VSAM. *VSAM(Virtual Storage Access Method)* 참조.

VSAM(Virtual Storage Access Method). 직접 액세스 장치에서 고정 길이와 가변 길이의 레코드를 직접 또는 순차적으로 처리하기 위한 액세스 방법. VSAM 데이터 세트나 파일의 레코드는 키 필드(키 순서)에 의해 논리적 순서대로, 데이터 세트나 파일에서 작성된 물리적 순서(입력 순서)대로, 또는 상대적 레코드 번호에 의해 구성될 수 있습니다.

VTAM. *VTAM(Virtual Telecommunication Access Method)* 참조.

VTAM(Virtual Telecommunications Access Method). OS/390 환경에서, 통신과 SNA 네트워크에서의 데이터 흐름을 제어하는 IBM 사용권 부여 프로그램.

W

WLM 응용프로그램 환경. 하나 이상의 저장 프로시듀어와 연관되는 MVS 워크로드 관리 프로그램 속성. WLM 응용프로그램 환경은 주어진 OS/390용 DB2 UDB 저장 프로시듀어가 수행되는 주소를 판별합니다.

WTO. *WTO(Write To Operator)* 참조.

WTOR. 응답이 있는 WTO(Write To Operator).

WTO(Write To Operator). 운영자에게 정정해야 하는 오류와 비정상적 시스템 상태를 알리는 메시지를 시스템 콘솔 운영자에게 보낼 수 있게 하는 선택적 사용자 코딩 서비스.

X

XCF. *상호 시스템 결합 기능* 참조.

XID. 교환국 ID(exchange station ID).

XRF. *확장 복구 기능* 참조.

부록R. DB2 라이브러리 사용

DB2 Universal Database 라이브러리는 온라인 도움말, 책(PDF 및 HTML) 및 샘플 프로그램이 HTML 형식으로 구성됩니다. 이 절에서는 제공되는 정보 및 액세스하는 방법을 설명합니다.

제품 정보에 온라인으로 액세스하려면, 정보 센터를 이용할 수 있습니다. 1616 페이지의 『정보 센터로 정보에 액세스』에서 자세한 내용을 참조하십시오. 웹에서 타스크 정보, DB2 책, 문제점 해결 정보, 샘플 프로그램 및 DB2 정보를 열람할 수 있습니다.

DB2 PDF 파일 및 인쇄된 책

DB2 정보

다음의 테이블은 DB2 책을 4개의 범주로 나눕니다.

DB2 안내 및 참조 정보

이 책에는 모든 플랫폼에 공통적인 DB2 정보가 들어 있습니다.

DB2 설치 및 구성 정보

이 책에는 특정 플랫폼의 DB2를 위한 것입니다. 예를 들어, OS/2, Windows 및 UNIX 플랫폼에서의 DB2용으로 각각 다른 *빠른 시작* 책이 있습니다.

플랫폼간 샘플 프로그램(HTML)

이 샘플들은 응용프로그램 개발 클라이언트와 함께 설치된 샘플 프로그램의 HTML 버전입니다. 이들은 단지 정보용으로서 실제 프로그램을 대체하지는 않습니다.

릴리스 정보

이러한 파일에는 DB2 책에 포함될 수 없었던 최신 정보가 포함되어 있습니다.

설치 매뉴얼, 릴리스 정보 및 자습서는 제품 CD-ROM의 HTML 디렉토리에서 볼 수 있습니다. 대부분의 책은 단지 보기용으로 제품 CD-ROM에서 HTML 형식으로 제공되고 보기와 인쇄용으로 제품 CD-ROM에서 PDF 형식으로 제공됩니다. 또한 IBM에서 인쇄된 책을 주문하려면 1611 페이지의 『인쇄된 책 주문』에서 자세한 내용을 참조하십시오. 다음 테이블에는 주문할 수 있는 책을 보여줍니다.

OS/2 및 Windows 플랫폼에서는 sqllib\doc\html 디렉토리에 HTML 파일을 설치할 수 있습니다. DB2 정보는 여러 나라 언어로 번역되었습니다. 하지만, 모든 정보가 모든 나라의 언어로 번역된 것은 아닙니다. 정보가 특정 나라의 언어로 사용할 수 없을 경우에는 영문으로 제공됩니다.

UNIX 플랫폼에서는 doc/%*L*/html 디렉토리에 여러 나라 언어 버전의 HTML 파일을 설치할 수 있습니다. 여기서 *%L*은 해당 언어의 로케일을 나타냅니다. *빠른 시작* 책에서 보다 자세한 내용을 참조하십시오.

다음의 여러 가지 방법으로 DB2 책을 구하고 정보를 액세스할 수 있습니다.

- 1615 페이지의 『정보 온라인 보기』
- 1619 페이지의 『정보 온라인 검색』
- 1611 페이지의 『인쇄된 책 주문』
- 1611 페이지의 『PDF 책 인쇄』

표 147. DB2 정보

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
	DB2 안내 및 참조 정보		
관리 안내서	*관리 안내서: 계획*에서는 데이터베이스의 개념에 대한 개요, 논리적 또는 물리적인 데이터베이스 설계와 같은 설계에 대한 정보 그리고 고가용성에 대한 정보를 제공합니다. *관리 안내서: 구현*에서는 사용자의 설계, 데이터베이스 액세스, 감사, 백업 및 복구와 같은 구현에 대한 정보를 제공합니다. *관리 안내서: 성능*에서는 데이터베이스의 환경, 응용프로그램 성능 평가 및 성능 조정에 대한 정보를 제공합니다. 사용자는 문서 번호 SBOF-8934를 사용하여 세 권으로 된 *관리 안내서* 책을 주문할 수 있습니다.	SA30-0990 db2d1x70 SA30-0988 db2d2x70 SA30-0989 db2d3x70	db2d0
Administrative API Reference	데이터베이스를 관리하는 데 사용할 수 있는 DB2 API와 데이터 구조에 대해 설명합니다. 또한 응용프로그램에서 API를 호출하는 방법을 설명합니다.	SC09-2947 db2b0x70	db2b0
응용프로그램 빌드 안내서	환경 설정 정보와 Windows에서 DB2 응용프로그램을 컴파일, 링크 및 수행하기 위한 지침이 단계별로 제공되어 있습니다.	SA30-0991 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	DB2 Universal Database 제품을 사용할 때 발생할 수 있는 APPC, CPI-C 및 SNA 센스 코드에 관한 일반 정보를 제공합니다. HTML 형식으로만 사용할 수 있습니다.	문서 번호가 없습니다. db2apx70	db2ap
응용프로그램 개발 안내서	Embedded SQL 또는 Java(JDBC 및 SQLJ)를 사용하여 DB2 데이터베이스를 액세스하는 응용프로그램을 개발하는 방법을 설명합니다. 저장 프로시듀어 작성, 사용자 정의 함수 작성, 사용자 정의 유형 작성, 트리거 사용, 파티션된 환경 또는 연합 시스템에서 응용프로그램을 개발하는 등의 다양한 주제가 다루어집니다.	SA30-0992 db2a0x70	db2a0

표 147. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
CLI Guide and Reference	DB2 콜 레벨 인터페이스와 Microsoft ODBC 스펙과 호환 가능한 호출 가능 SQL 인터페이스를 사용하여 DB2 데이터베이스에 액세스하는 응용프로그램의 개발 방법에 대해 설명합니다.	SC09-2950 db2l0x70	db2l0
Command Reference	명령행 프로세서를 사용하는 방법을 설명하고 데이터베이스를 관리하기 위해 사용할 수 있는 DB2 명령을 설명합니다.	SC09-2951 db2n0x70	db2n0
연결성 보충 설명서	AS/400용 DB2, OS/390용 DB2, MVS용 DB2 또는 VM용 DB2를 DB2 Universal Database 서버와의 DRDA 응용프로그램 리퀘스터로 사용하는 방법에 대한 참조 정보 및 설치 정보를 제공합니다. 또한 DB2 Connect AR(응용프로그램 리퀘스터)과 함께 DRDA AS(응용프로그램 서버)를 사용하는 방법에 대해서도 상세히 설명합니다. HTML 및 PDF 형식으로만 사용할 수 있습니다.	문서 번호가 없습니다. db2h1x70	db2h1
데이터 이동 유틸리티 안내 및 참조서	Import, Export, Load, AutoLoader 및 DPROP와 같이 데이터 이동을 용이하게 해 주는 DB2 UDB 유틸리티의 사용 방법에 대해 설명합니다.	SA30-0994 db2dmx70	db2dm
Data Warehouse Center 관리 안내서	Data Warehouse Center를 사용하여 데이터 웨어하우스를 구축 및 유지보수하는 방법을 제공합니다.	SA30-1000 db2ddx70	db2dd
Data Warehouse Center 응용프로그램 통합 안내서	프로그래머들이 Data Warehouse Center 및 Information Catalog Manager를 응용프로그램과 통합하는 데 도움을 주는 정보를 제공합니다.	SA30-1001 db2adx70	db2ad
DB2 Connect 사용자 안내서	DB2 Connect 제품에 대한 개념, 프로그래밍 및 일반 사용 정보를 제공합니다.	SA30-0993 db2c0x70	db2c0
DB2 Query Patroller Administration Guide	DB2 Query Patroller 시스템의 조작 개요, 특정 조작 및 관리 정보, 관리 그래픽 사용자 인터페이스 유틸리티에 대한 타스크 정보를 제공합니다.	SC09-2958 db2dwx70	db2dw

표 147. DB2 정보 (계속)

이름	설명	문서 번호 **PDF 파일 이름**	**HTML 디렉토리**
DB2 Query Patroller User's Guide	DB2 Query Patroller의 도구 및 함수를 사용하는 방법을 설명합니다.	SC09-2960 db2wwx70	db2ww
용어집	DB2에서 사용되는 용어와 그 구성요소에 대한 정의를 제공합니다. HTML 형식과 *SQL 참조서*에서 사용할 수 있습니다.	문서 번호가 없습니다. db2t0x70	db2t0
Image, Audio 및 Video Extenders 관리 및 프로그래밍	DB2 Extender에 대한 일반적인 정보와 이미지, 오디오 및 비디오(IAV)의 관리 및 구성에 대한 정보 그리고 IAV Extenders를 사용한 프로그램에 대한 정보를 제공합니다. 여기에는 참조 정보, 진단 정보(메시지 포함) 및 샘플도 들어 있습니다.	SA30-1043 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	정보 카탈로그 관리에 대한 지침을 제공합니다.	SC26-9995 db2dix70	db2di
Information Catalog Manager Programming Guide and Reference	Information Catalog Manager에 대한 아키텍처 인터페이스에 대한 정의를 제공합니다.	SC26-9997 db2bix70	db2bi
Information Catalog Manager 사용자 안내서	Information Catalog Manager 사용자 인터페이스 사용에 대한 정보를 제공합니다.	SA30-1002 db2aix70	db2ai
설치 및 구성 보충 설명서	플랫폼 특정 DB2 클라이언트의 플랜, 설치 및 설정에 대해 설명합니다. 또한 바인딩, 클라이언트 및 서버 통신의 설정, DB2 GUI 도구, DRDA AS, 분산 설치 및 이종 데이터 소스에 대한 분산 요구와 액세스 방식의 구성에 대한 정보가 들어 있습니다.	GA30-0975 db2iyx70	db2iy
메시지 참조서	DB2, Information Catalog Manager 및 Data Warehouse Center에서 발행하는 메시지와 코드를 나열하고 수행해야 할 조치에 대해 설명합니다. 문서 번호(SBOF-8932)를 사용하여 두 권으로 된 메시지 참조서 책을 모두 주문할 수 있습니다.	볼륨 1 GA30-0986 db2m1x70 볼륨 2 GA30-0987 db2m2x70	db2m0

표 147. DB2 정보 (계속)

이름	설명	문서 번호 **PDF 파일 이름**	**HTML 디렉토리**
OLAP Integration Server Administration Guide	OLAP 통합 서버의 관리 프로그램 구성요소를 사용하는 방법을 설명합니다.	SC27-0787 db2dpx70	n/a
OLAP Integration Server Metaoutline User's Guide	표준 OLAP Metaoutline 인터페이스(Metaoutline Assistant가 아닌)를 사용하여 OLAP Metaoutlines을 작성하고 사용하는 방법을 설명합니다.	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	표준 OLAP 모델 인터페이스(Model Assistant가 아닌)를 사용하여 OLAP 모델을 작성하는 방법을 설명합니다.	SC27-0783 db2lpx70	n/a
OLAP 설치 및 사용자 안내서	OLAP Starter Kit에 대한 구성 및 설치 정보를 제공합니다.	SA30-1074 db2ipx70	db2ip
Excel용 OLAP Spreadsheet Add-in 사용자 안내서	Excel 스프레드시트 프로그램을 사용하여 OLAP 데이터를 분석하는 방법을 설명합니다.	SA30-0564 db2epx70	db2ep
Lotus 1-2-3용 OLAP Spreadsheet Add-in 사용자 안내서	Lotus 1-2-3 스프레드시트 프로그램을 사용하여 OLAP 데이터를 분석하는 방법을 설명합니다.	SA30-0565 db2tpx70	db2tp
복제 안내 및 참조서	DB2와 함께 제공된 IBM 복제 도구에 관한 플랜, 구성, 관리 및 사용 정보를 제공합니다.	SA30-1003 db2e0x70	db2e0
Spatial Extender 사용자 안내 및 참조서	Spatial Extender 설치, 구성, 관리, 프로그래밍 및 문제 해결에 대한 정보를 제공합니다. 또한 공간 데이터 개념에 대한 설명을 제공하고 Spatial Extender에만 고유하게 적용되는 참조 정보(메시지 및 SQL)를 제공합니다.	SA30-1045 db2sbx70	db2sb
SQL 시작하기	SQL 개념을 소개하고, 많은 구조와 타스크에 관한 예를 보여줍니다.	SA30-0996 db2y0x70	db2y0
SQL 참조서, 볼륨 1 및 볼륨 2	SQL 구문, 의미 그리고 언어 규칙에 대해 설명합니다. 또한 릴리스 간 비호환성, 제품 제한사항 및 카탈로그 뷰에 대한 정보도 들어 있습니다. SBOF-8933 문서 번호를 사용하여 *SQL 참조서*를 주문할 수 있습니다.	볼륨 1 SA30-0997 db2s1x70 볼륨 2 SA30-0998 db2s2x70	db2s0

표 147. DB2 정보 (계속)

이름	설명	문서 번호 **PDF 파일 이름**	**HTML 디렉토리**
시스템 모니터 안내 및 참조서	데이터베이스와 데이터베이스 관리 프로그램에 관한 여러 종류의 정보를 수집하는 방법에 대해 설명합니다. 이 책은 데이터베이스 활동을 이해하고, 성능을 향상시키고, 문제점의 원인을 판별하기 위한 정보를 사용하는 방법을 설명합니다.	SA30-0995 db2f0x70	db2f0
Text Extender 관리 및 프로그래밍	DB2 Extenders에 관한 일반적인 정보와 Text Extenders 관리 및 구성에 관한 정보, Text Extenders를 사용한 프로그래밍에 관한 정보를 제공합니다. 여기에는 참조 정보, 진단 정보(메시지 포함) 및 샘플도 들어 있습니다.	SA30-1044 desu9x70	desu9
문제점 해결 안내서	오류의 출처를 판별하고 문제점으로부터 회복하고, DB2 고객 서비스와 상담하여 진단 도구를 사용하는 것을 도와줍니다.	GA30-0704 db2p0x70	db2p0
새로운 기능	DB2 Universal Database, 버전 7의 새로운 특성, 기능 및 향상된 내용을 설명합니다.	SA30-0999 db2q0x70	db2q0
	DB2 설치 및 구성 정보		
OS/2 및 Windows용 DB2 Connect Enterprise Edition 빠른 시작	OS/2 및 Windows 32 비트 운영 체제에서 DB2 Connect Enterprise Edition에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0974 db2c6x70	db2c6
UNIX용 DB2 Connect Enterprise Edition 빠른 시작	UNIX 기반 플랫폼에서의 DB2 Connect Enterprise Edition에 대한 플랜, 이주, 설치, 구성 및 타스크 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0973 db2cyx70	db2cy
DB2 Connect Personal Edition 빠른 시작	OS/2 및 Windows 32 비트 운영 체제에서 DB2 Connect Personal Edition에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 모든 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0981 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings for Linux	지원되는 모든 Linux에서 DB2 Connect Personal Edition에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다.	GC09-2962 db2c4x70	db2c4

표 147. DB2 정보 (계속)

이름	설명	문서 번호 **PDF 파일 이름**	**HTML 디렉토리**
DB2 Data Links Manager 빠른 시작	AIX 및 Windows 32 비트 운영 체제용 DB2 Data Links Manager에 대한 플랜, 설치, 구성 및 타스크 정보를 제공합니다.	GA30-0980 db2z6x70	db2z6
UNIX용 DB2 Enterprise - Extended Edition 빠른 시작	UNIX 기반 플랫폼에서의 DB2 Enterprise - Extended Edition 플랜, 설치 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0978 db2v3x70	db2v3
Windows용 DB2 Enterprise - Extended Edition 빠른 시작	Windows 32 비트 운영 체제용 DB2 Enterprise - Extended Edition에 관한 플랜, 설치 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0977 db2v6x70	db2v6
OS/2용 DB2 빠른 시작	OS/2 운영 체제에서의 DB2 Universal Database에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0982 db2i2x70	db2i2
UNIX용 DB2 빠른 시작	UNIX 기반 플랫폼에서의 DB2 Universal Database에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0984 db2ixx70	db2ix
Windows용 DB2 빠른 시작	Windows 32 비트 운영 체제에서 DB2 Universal Database에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0985 db2i6x70	db2i6
DB2 Personal Edition 빠른 시작	OS/2 및 Windows 32 비트 운영 체제에서의 DB2 Universal Database Personal Edition에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다.	GA30-0983 db2i1x70	db2i1
DB2 Personal Edition Quick Beginnings for Linux	지원되는 모든 Linux에서 DB2 Universal Database Personal Edition에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다.	GC09-2972 db2i4x70	db2i4
DB2 Query Patroller 설치 안내서	DB2 Query Patroller에 관한 설치 정보를 제공합니다.	GA30-0976 db2iwx70	db2iw

표 147. DB2 정보 (계속)

이름	설명	문서 번호 **PDF 파일 이름**	**HTML 디렉토리**
DB2 Warehouse Manager 설치 안내서	웨어하우스 에이전트, 웨어하우스 변환기 및 Information Catalog Manager에 관한 설치 정보를 제공합니다.	GA30-1027 db2idx70	db2id
	플랫폼간 샘플 프로그램(HTML)		
샘플 프로그램(HTML)	DB2가 지원하는 모든 플랫폼에서 프로그래밍 언어에 대한 샘플 프로그램이 HTML 형식으로 제공됩니다. 이 샘플 프로그램은 정보용으로만 제공됩니다. 모든 샘플을 모든 프로그래밍 언어로 사용할 수 있는 것은 아닙니다. HTML 샘플은 DB2 응용프로그램 개발 클라이언트가 설치될 때에 사용할 수 있습니다. 프로그램에 대한 정보는 *응용프로그램 빌드 안내서*에서 자세한 내용을 참조하십시오.	문서 번호가 없습니다.	db2hs
	릴리스 정보		
DB2 Connect 릴리스 정보	DB2 Connect 책에는 포함될 수 없었던 최신 정보를 제공합니다.	#2를 참조하십시오.	db2cr
DB2 설치 정보	DB2 책에는 포함될 수 없었던 최신 설치 정보를 제공합니다.	제품 CD-ROM에서만 사용할 수 있습니다.	
DB2 릴리스 정보	DB2 책에는 포함될 수 없었던 모든 DB2 제품 및 기능에 대한 최신 정보를 제공합니다.	#2를 참조하십시오.	db2ir

주:

1. 파일 이름의 6번째 자리에 있는 문자 *x*는 책의 언어 버전을 나타냅니다. 예를 들면, 파일 이름 `db2d0e70`은 *관리 안내서* 책의 영문 버전을 나타내며 `db2d0k70`은 같은 책의 한글 버전을 나타냅니다. 다음 문자는 언어 버전을 나타내기 위해 파일 이름의 6번째 자리에 사용됩니다.

언어	식별자
브라질 포르투갈어	b
불가리아어	u
체코어	x
덴마크어	d
네덜란드어	q
영어	e

핀란드어	y
프랑스어	f
독일어	g
그리스어	a
헝가리어	h
이탈리아어	i
일본어	j
한글	k
노르웨이어	n
폴란드어	p
포르투갈어	v
러시아어	r
중국어	c
슬로베니아어	l
스페인어	z
스웨덴어	s
대만어	t
터키어	m

2. DB2 책에 포함되어 있지 않을 수 있는 최신 정보는 릴리스 정보에서 HTML 형식과 ASCII 파일로 사용할 수 있습니다. HTML 버전은 정보 센터와 제품 CD-ROM에서 사용할 수 있습니다. ASCII 파일을 보려면 다음을 참조하십시오.
   - UNIX 기반 플랫폼의 경우에는 Release.Notes 파일을 참조하십시오. 이 파일은 DB2DIR/Readme/*%L* 디렉토리에 있으며 여기서, *%L*은 로케일 이름이고 DB2DIR은 다음과 같습니다.
     - AIX에서는 /usr/lpp/db2_07_01
     - HP-UX, PTX, Solaris 및 Silicon Graphics IRIX에서는 /opt/IBMdb2/V7.1
     - Linux에서는 /usr/IBMdb2/V7.1
   - 다른 플랫폼의 경우에는 RELEASE.TXT 파일을 참조하십시오. 이 파일은 제품이 설치된 디렉토리에 있습니다. OS/2 플랫폼에서는 **IBM DB2** 폴더를 더블 클릭하고 **릴리스 정보** 아이콘을 더블 클릭할 수 있습니다.

PDF 책 인쇄

책의 사본을 원하는 경우 DB2 책 CD-ROM에 있는 PDF 파일을 인쇄할 수 있습니다. Adobe Acrobat Reader를 사용하여 책 전체나 특정 페이지를 인쇄할 수 있습니다. 라이브러리에 있는 각 책의 파일 이름에 대해서는 1603 페이지의 표147에서 자세한 내용을 참조하십시오.

Adobe 웹 사이트인 http://www.adobe.com에서 Adobe Acrobat Reader의 최신 버전을 얻을 수 있습니다.

PDF 파일은 파일 확장자가 PDF로서 DB2 책 CD-ROM에 들어 있습니다. PDF 파일을 액세스하려면,

1. DB2 책 CD-ROM을 삽입하십시오. UNIX 기반의 플랫폼에서는 DB2 책 CD-ROM을 마운트해야 합니다. 마운트 절차에 대해서는 *빠른 시작* 책에서 자세한 내용을 참조하십시오.
2. Acrobat Reader를 시작하십시오.
3. 다음 위치에서 원하는 PDF 파일을 여십시오.
   - OS/2 및 Windows 플랫폼에서:

     x:\doc*language* 디렉토리. 여기서 *x*는 CD-ROM 드라이브를 나타내며 *language*는 사용자 언어를 나타내는 2문자 국가 코드를 나타냅니다. 예를 들면 영문인 경우에는 EN입니다.
   - UNIX 기반 플랫폼에서:

     /cdrom/doc/*%L* 디렉토리. 여기서 */cdrom*은 CD-ROM의 마운트 위치이고 *%L*은 원하는 로케일의 이름입니다.

또한 PDF 파일을 CD-ROM에서 지역이나 네트워크로 파일을 복사하고 거기서 읽을 수도 있습니다.

인쇄된 책 주문

인쇄된 DB2 책은 책 주문 번호(SBOF)를 사용하여 세트나 낱권으로 주문할 수 있습니다. 인쇄본을 주문하려면, IBM 협력업체 또는 영업 대표에게 문의하십시오. 또한 웹 사이트 http://www.elink.ibmlink.ibm.com/pbl/pbl에서도 책을 주문할 수 있습니다.

두 종류의 책 세트를 사용할 수 있습니다. SBOF-8935는 DB2 Warehouse Manager에 대한 참조 및 사용에 관한 정보를 제공합니다. SBOF-8931은 다른 모든 DB2 Universal Database 제품과 특징에 대한 참조 및 사용 정보를 제공합니다. 각 SBOF의 내용은 다음 테이블에 나열되어 있습니다.

표 148. 인쇄된 책 주문

SBOF 번호	**포함된 책**	
SBOF-8931	• 관리 안내서: 계획 • 관리 안내서: 구현 • 관라 안내서: 성능 • Administrative API Reference • 응용프로그램 빌드 안내서 • 응용프로그램 개발 안내서 • CLI Guide and Reference • Command Reference • 데이터 이동 유틸리티 안내 및 참조서 • Data Warehouse Center 관리 안내서 • Data Warehouse Center 응용프로그램 통합 안내서 • DB2 Connect 사용자 안내서 • 설치 및 구성 보충 설명서 • Image, Audio 및 Video Extenders 관리 및 프로그래밍 • 메시지 참조서, 볼륨 1 및 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP 설정 및 사용자 안내서 • Excel용 OLAP Spreadsheet Add-in 사용자 안내서 • Lotus 1-2-3용 OLAP Spreadsheet Add-in 사용자 안내서 • 복제 안내 및 참조서 • Spatial Extender 관리 및 프로그래밍 안내서 • SQL 시작하기 • SQL 참조서, 볼륨 1 및 2 • 시스템 모니터 안내 및 참조서 • Text Extender 관리 및 프로그래밍 • 문제점 해결 안내서 • 새로운 기능
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager 사용자 안내서 • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

DB2 온라인 문서

온라인 도움말 액세스

온라인 도움말은 모든 DB2 구성요소에서 사용할 수 있습니다. 다음의 테이블에서는 다양한 도움말 유형을 설명합니다.

도움말의 유형	목차	액세스하는 방법
명령 도움말	명령행 처리기 내의 명령 구문을 설명합니다.	대화식 모드인 명령행 처리기에서, 다음을 입력하십시오. `? command` 여기서, *command*는 키워드이거나 전체 명령입니다. 예를 들어, `? catalog`는 모든 CATALOG 명령에 대한 도움말을 표시하고, `? catalog database`는 CATALOG DATABASE 명령에 대한 도움말을 표시합니다.
클라이언트 구성 지원 프로그램 도움말 **명령 센터 도움말** **제어 센터 도움말** **Data Warehouse Center 도움말** **이벤트 분석기 도움말** **Information Catalog Manager 도움말** **위성 관리 센터 도움말** **스크립트 센터 도움말**	창 또는 노트북에서 수행할 수 있는 타스크를 설명합니다. 도움말은 알아야 할 개요와 전제조건 정보를 포함하고, 창 또는 노트북 제어를 사용하는 방법을 설명합니다.	창이나 노트북에서, **도움말** 버튼을 누르거나 **F1** 키를 누르십시오.

도움말의 유형	목차	액세스하는 방법

메시지 도움말	메시지의 원인과 사용자가 취해야 할 조치를 설명합니다.	대화식 모드인 명령행 처리기에서, 다음을 입력하십시오.
		`? XXXnnnnn`
		여기서, *XXXnnnnn*은 유효한 메시지 식별자입니다.
		예를 들어, `? SQL30081`은 SQL30081 메시지에 대한 도움말을 표시합니다.
		한 번에 한 화면씩 메시지 도움말을 보려면, 다음을 입력하십시오.
		`? XXXnnnnn | more`
		파일에 메시지 도움말을 저장하려면, 다음을 입력하십시오.
		`? XXXnnnnn > filename.ext`
		여기서, *filename.ext*는 메시지 도움말을 저장하려는 파일입니다.

SQL 도움말	SQL문의 구문을 설명합니다.	대화식 모드인 명령행 처리기에서, 다음을 입력하십시오. `help statement` 여기서, *statement*는 SQL문입니다. 예를 들어, help SELECT는 SELECT문에 대한 도움말을 표시합니다. **주:** SQL 도움말은 UNIX 기반 플랫폼에서 사용할 수 없습니다.
SQLSTATE 도움말	SQL 상태 및 클래스 코드를 설명합니다.	대화식 모드인 명령행 처리기에서, 다음을 입력하십시오. `? sqlstate` 또는 `? class code` 여기서, *sqlstate*는 유효한 5자리 숫자로 된 SQL 상태이고 *class code*는 SQL 상태의 처음 2자리 숫자입니다. 예를 들어, `? 08003`은 08003 SQL 상태에 대한 도움말을 표시하고, `? 08`은 08 클래스 코드에 대한 도움말을 표시합니다.

정보 온라인 보기

이 제품에 들어 있는 책은 HTML(Hypertext Markup Language) 소프트카피 형식으로 제공됩니다. 소프트카피는 정보를 검색할 수 있게 하고 관련된 정보로 링크하는 하이퍼텍스트를 제공합니다. 또한, 사이트에서 라이브러리를 공유하는 것도 더 쉬워집니다.

HTML 버전 3.2 스펙을 따르는 브라우저로 온라인 책 또는 샘플 프로그램을 볼 수 있습니다.

온라인 책 또는 샘플 프로그램을 보려면:

- DB2 관리 도구를 수행할 경우, 정보 센터를 사용하십시오.
- 브라우저에서, **파일 —>페이지 열기**를 클릭하십시오. 열린 페이지에 DB2 정보에 대한 설명과 링크가 들어 있습니다.
  - UNIX 기반 플랫폼에서는 다음과 같은 페이지를 여십시오.

    INSTHOME/sqllib/doc/%*L*/html/index.htm

    여기서 %*L*은 로케일 이름입니다.
  - 다른 플랫폼에서는 다음과 같은 페이지를 여십시오.

    sqllib\doc\html\index.htm

    경로는 DB2가 설치된 드라이브에 있습니다.

    정보 센터를 설치하지 않은 경우, **DB2 정보** 아이콘을 더블 클릭하여 페이지를 열 수 있습니다. 사용하는 시스템에 따라, 주 제품 폴더나 Windows 시작 메뉴에 아이콘이 있습니다.

Netscape 브라우저 설치

웹 브라우저를 설치하지 않은 경우, 제품 상자에 있는 Netscape CD-ROM에서 Netscape를 설치할 수 있습니다. 설치하는 방법에 대한 자세한 지시 사항은 다음을 수행하십시오.

1. Netscape CD-ROM을 삽입하십시오.
2. UNIX 기반의 플랫폼에서는 CD-ROM을 마운트해야 합니다. 마운트 절차에 대해서는 *빠른 시작* 책에서 자세한 내용을 참조하십시오.

3. 설치 지침서는 CDNAV*nn*.txt 파일을 참조하십시오. 여기서, *nn*은 2문자로 된 언어 식별자입니다. 파일은 CD-ROM의 루트 디렉토리에 있습니다.

정보 센터로 정보에 액세스

정보 센터는 DB2 제품 정보로의 빠른 액세스를 제공합니다. 정보 센터는 DB2 관리 도구를 사용할 수 있는 모든 플랫폼에서 사용할 수 있습니다.

정보 센터 아이콘을 더블 클릭하여 정보 센터를 열 수 있습니다. 사용하는 시스템에 따라 아이콘은 주 제품 폴더나 Windows **시작** 메뉴의 정보 폴더에 있습니다.

또한 DB2 Windows 플랫폼에서 도구 모음이나 **도움말** 메뉴를 사용하여 정보 센터를 액세스할 수 있습니다.

정보 센터는 6개 유형의 정보를 제공합니다. 적당한 탭을 클릭하여 그 유형에서 지원하는 주제를 보십시오.

타스크 DB2를 사용하여 수행할 수 있는 키 타스크.

참조 키워드, 명령 및 API와 같은 DB2 참조 정보.

책 DB2 책.

문제점 해결 오류 메시지의 종류와 복구 조치.

샘플 프로그램 DB2 응용프로그램 개발 클라이언트와 함께 제공되는 샘플 프로그램. DB2 응용프로그램 개발 클라이언트를 설치하지 않은 경우, 이 탭은 표시되지 않습니다.

웹 월드 와이드 웹에서의 DB2 정보. 이 정보에 액세스하려면, 사용자의 시스템으로부터 웹으로의 연결이 있어야 합니다.

목록 중 하나에서 항목을 선택할 때, 정보 센터는 정보를 표시하기 위해 표시기를 시작합니다. 표시기는 사용자가 선택하는 정보의 종류에 따라, 시스템 도움말 표시기, 편집기 또는 웹브라우저가 될 수 있습니다.

정보 센터는 찾기 기능을 제공하므로 목록을 찾지 않고도 특정 주제를 찾을 수 있습니다.

전체 텍스트 검색을 위해서는 **DB2 온라인 정보 검색** 검색 양식으로 연결된 정보 센터의 하이퍼텍스트 링크를 따라 검색하십시오.

HTML 검색 서버는 보통 자동으로 시작됩니다. HTML 정보에서 검색 기능이 작동하지 않으면, 다음 방법 중 하나를 사용하여 검색 서버를 시작할 수 있습니다.

Windows의 경우:

시작을 클릭하고 **프로그램 —> IBM DB2 —> 정보 —> HTML 검색 서버 시작**을 선택하십시오.

OS/2 경우:

OS/2용 DB2 폴더를 더블 클릭하고 **HTML 검색 서버 시작** 아이콘을 더블 클릭하십시오.

HTML 정보를 검색하면서 다른 문제가 생길 경우, 릴리스 정보를 참조하십시오.

주: 검색 기능은 Linux, PTX 및 Silicon Graphics IRIX 환경에서는 작동하지 않습니다.

DB2 마법사 사용

마법사는 한 번에 한 단계씩 각 작업을 수행하게 함으로써 특정 관리 타스크를 완료하는 데 도움을 줍니다. 마법사는 제어 센터 및 클라이언트 구성 지원 프로그램을 통해 사용할 수 있습니다. 다음 테이블에서는 마법사를 나열하고 그 기능을 설명합니다.

주: 데이터베이스 작성, 색인 작성, 다중 사이트 갱신 구성 및 성능 구성 마법사는 파티션된 데이터베이스 환경에서 사용할 수 있습니다.

마법사	도움 대상	액세스하는 방법
데이터베이스 추가	클라이언트 워크스테이션의 데이터베이스를 카탈로그화합니다.	클라이언트 구성 지원 프로그램에서 **추가**를 클릭하십시오.
데이터베이스 백업	백업 계획을 결정하고, 작성하고, 일정을 세웁니다.	제어 센터에서 백업하려는 데이터베이스를 마우스의 오른쪽 버튼으로 클릭한 다음 **백업 —> 마법사를 사용한 데이터베이스 백업**을 선택하십시오.
다중 사이트 갱신 구성	다중 사이트 갱신, 분선 트랜잭션 또는 2 단계 확약을 구성합니다.	제어 센터에서 **데이터베이스** 폴더를 마우스의 오른쪽 버튼으로 클릭하고 **다중 사이트 갱신**을 선택하십시오.

마법사	도움 대상	액세스하는 방법
데이터베이스 작성	데이터베이스를 작성한 다음, 몇 가지 기본적인 구성 타스크를 수행합니다.	제어 센터에서 **데이터베이스** 폴더를 마우스의 오른쪽 버튼으로 클릭한 다음 **작성 —> 마법사를 사용한 데이터베이스 작성**을 선택하십시오.
테이블 작성	기본 데이터 유형을 선택한 다음, 테이블에 대한 기본 키를 작성합니다.	제어 센터에서 **테이블** 아이콘을 마우스의 오른쪽 버튼으로 클릭하고 **작성 —> 마법사를 사용한 테이블**을 선택하십시오.
테이블 공간 작성	새로운 테이블 공간을 작성합니다.	제어 센터에서 **테이블 공간** 아이콘을 마우스의 오른쪽 버튼으로 선택하고 **작성 —> 마법사를 사용한 테이블 공간**을 선택하십시오.
색인 작성	사용자의 모든 조회를 작성하고 삭제하기 위해 색인화합니다.	제어 센터에서 **색인** 아이콘을 마우스의 오른쪽 버튼으로 클릭하고 **작성 —> 마법사를 사용한 색인**을 선택하십시오.
성능 구성	업무 요구조건에 맞게 구성 매개변수를 갱신하여 데이터베이스의 성능을 조정합니다.	제어 센터에서 성능을 조정하려는 데이터베이스를 마우스의 오른쪽 버튼으로 클릭하고 **마법사를 사용한 성능 구성**을 선택하십시오. 파티션된 데이터베이스에 대해 데이터베이스 파티션 뷰로부터 성능을 조정하려는 첫번째 파티션을 마우스의 오른쪽 버튼으로 클릭하고 **마법사를 사용한 성능 구성**을 선택하십시오.
데이터베이스 복원	실패 후에 데이터베이스를 복구합니다. 사용할 백업 위치 및 재작동할 로그 기록을 이해하는 데 도움을 줍니다.	제어 센터에서 복원하려는 데이터베이스를 마우스의 오른쪽 버튼으로 클릭한 다음 **복원 —> 마법사를 사용한 데이터베이스**를 선택하십시오.

문서 서버 설정

기본 값으로 DB2 정보는 지역 시스템에 설치됩니다. 이는 DB2 정보에 액세스해야 하는 모든 사람이 동일한 파일을 설치해야 함을 의미합니다. DB2 정보를 한 위치에 저장하려면, 다음과 같이 하십시오.

1. 지역 시스템의 \sqllib\doc\html에 있는 모든 파일과 서브디렉토리를 웹 서버로 복사하십시오. 각 책은 책을 구성하는 데 필요한 모든 HTML 및 GIF 파일이 들어 있는 서브디렉토리를 가집니다. 디렉토리 구조가 변경되지 않게 하십시오.

2. 새로운 위치에 있는 파일을 찾도록 웹 서버를 구성하십시오. 보다 자세한 정보는 *설치 및 구성 보충 설명서*의 부록 NetQuestion을 참조하십시오.
3. Java 버전의 정보 센터를 이용하는 경우, 모든 HTML 파일에 대한 기본 URL을 지정할 수 있습니다. 책 목록에 대해서는 URL을 사용해야 합니다.
4. 책 파일을 열람할 수 있게 되면, 다음과 같이 자주 열람하는 주제 항목에 대해서는 북마크를 설정할 수 있습니다. 다음의 페이지들을 북마크로 설정해 두면 도움이 될 것입니다.
   - 책 목록
   - 자주 이용하는 책의 목차
   - ALTER TABLE 주제와 같은 자주 참조하는 항목
   - 검색 양식

DB2 Universal Database 온라인 문서 파일을 중앙 시스템에서 제공하는 방법에 대한 정보를 보려면 *설치 및 구성 보충 설명서*의 부록 NetQuestion을 참조하십시오.

정보 온라인 검색

HTML 파일에서 정보를 찾으려면, 다음 방법 중 하나를 사용하십시오.

- 맨 위 프레임에서 **검색**을 클릭하십시오. 특정 주제를 찾으려면 검색 양식을 사용하십시오. 이 기능은 Linux, PTX 또는 Silicon Graphics IRIX 환경에서는 사용할 수 없습니다.
- 맨 위 프레임에서 **색인**을 클릭하십시오. 책에서 특정 주제를 찾으려면 색인을 사용하십시오.
- 책에서 특정 주제를 찾으려면 목차나 도움말의 색인 또는 HTML 책을 표시하고 웹 브라우저의 찾기 기능을 사용하십시오.
- 특정 주제로 빨리 리턴하려면 웹 브라우저의 북마크 기능을 사용하십시오.
- 특정 주제를 찾으려면 정보 센터의 검색 기능을 사용하십시오. 1616 페이지의 『정보 센터로 정보에 액세스』에서 자세한 내용을 참조하십시오.

부록S. 주의사항

IBM은 이 책에서 논의된 제품, 서비스 또는 기능을 다른 나라에서는 제공하지 않을 수 있습니다. 현재 사용자가 사용할 수 있는 제품 및 서비스에 대한 정보는 해당 지역의 IBM 영업대표에게 문의하십시오. IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 반드시 IBM 제품, 프로그램 또는 서비스만을 사용해야 함을 의미하지는 않습니다. IBM의 지적 소유권을 침해하지 않는 기능상으로 동등한 타사의 제품, 프로그램 또는 서비스를 대신 사용할 수 있습니다. 그러나, 타사 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대한 특허를 보유하고 있거나 출원중일 수 있습니다. 이 책을 제공한다고 해서 그러한 특허에 대한 사용권까지를 부여하는 것은 아닙니다. 특허 사용권에 대한 문의는 다음 주소로 하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
지적 재산권부

2바이트(DBCS) 정보에 관한 사용권 문의는 사용자 국가의 IBM 지적 재산권부나 다음 주소로 서면 문의하십시오.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 사항은 영국이나 이 조항이 현지법과 상충되는 나라에는 적용되지 않습니다.
IBM에서는 이 책을 명시적 또는 암시적인 어떠한 종류의 보증없이『있는 그대로』제공하므로, 판매 가능성을 보장하거나 특정 목적에 적합한지 여부에 대해서는 책임질 수 없습니다. 일부 국가에서는 특정 거래의 명시적 또는 암시적인 보증을 부인하는 문장을 허용하지 않으므로, 이 사항이 사용자에게 적용되지 않을 수도 있습니다.

이 책에는 기술상 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 책의 내용은 정기적으로 변경되며, 이들 변경사항은 개정판에 통합됩니다. IBM은 사전 통지없이 언제든지 이 책에 설명된 제품과 프로그램을 개선 및 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 사용권자가 (i) 독립적으로 작성된 프로그램과 다른 프로그램(이 프로그램을 포함한) 사이의 정보 교환과 (2) 교환된 정보의 공동 사용을 목적으로 그 프로그램에 대한 정보를 원하는 경우, 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
소프트웨어 사업부

이러한 정보는 특정한 기간 및 조건하에 사용가능하며 어떤 경우에는 사용료를 지불해야 합니다.

이 책에 기술된 사용권 프로그램 및 이 프로그램에 사용가능한 모든 사용권 데이터는 IBM 고객 협약, IBM 국제 프로그래밍 사용권 협약 또는 이와 동등한 모든 협약 조건하에 IBM에서 제공됩니다.

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 제품과 관련된 정보는 해당 제품의 공급자, 공개 발표 또는 기타 공개적으로 사용가능한 소스에서 확보한 것입니다. IBM은 이들 제품을 검사하지 않았고 성능 상의 정확성, 호환성 또는 타사 제품과 관련된 기타 주장을 확인할 수 없습니다. 타사 제품의 성능에 관한 문제는 해당 제품의 공급자에게 제기되어야 합니다.

IBM이 제시하는 방향 또는 의도에 관한 어떠한 언급도 특별한 통지없이 변경될 수 있습니다.

이 정보는 일상적인 비지니스 처리에 사용되는 데이터와 보고서의 예가 들어 있을 수 있습니다. 보다 구체적으로 예를 나타내기 위해 특정 개인, 회사, 상표 또는 제품 이름이 언급되는 경우가 있습니다. 여기서 언급된 이름은 가상의 이름이며 실제 비지니스 업체가 사용하는 이름 및 주소와 유사하다면 우연인 것입니다.

사용권:

이 정보에는 여러 운영 체제에서 프로그래밍 소스 언어로 예제 응용프로그램이 들어 있을 수 있습니다. 사용자는 이들 예제 프로그램을 IBM에게 비용을 지급하지 않고 복사, 수정 및 분배할 수 있습니다. 이들 예제 프로그램은 모든 조건에서 철저하게 검사되지 않았습니다. 따라서, IBM은 이들 프로그램에 대해 어떠한 보증도 할 수 없습니다.

이들 예제 프로그램의 각각의 복사본이나 특정 부분은 다음과 같은 사용권 주의사항을 포함해야 합니다.

등록 상표

별표(*)로 표시된 다음의 용어는 미국 및 다른 나라의 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 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

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 미국 및 다른 나라의 Sun Microsystems, Inc.의 상표입니다.

Tivoli 및 NetView는 미국 및 다른 나라의 Tivoli Systems Inc.의 상표입니다.

UNIX는 미국 및 다른 나라의 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]

[라]

[마]

[바]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

[숫자]

A

B

C

D

E

F

G

H

I

J

O

P

Q

R

S

T

U

V

W

Y

[특수 문자]

IBM에 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 대표나 공인 IBM 소프트웨어 재판매업자에게문의하십시오.

미국에 사시는 분은 다음 번호 중 하나를 선택하여 전화하십시오.

- 고객 지원을 받으려면, 1-800-237-5511.
- 사용가능한 서비스 옵션을 알려면, 1-888-426-4343.

제품 정보

미국에 사시는 분은 다음 번호 중 하나를 선택하여 전화하십시오.

- 제품 주문이나 일반 정보를 얻으려면, 1-800-IBM-CALL (1-800-426-2255)이나 1-800-3IBM-OS2 (1-800-342-6672).
- 책에 대한 주문은 1-800-879-2755.

http://www.ibm.com/software/data/

DB2 월드 와이드 웹 페이지에는 새로운 소식, 제품 설명, 교육 일정 등에 관한 현재의 DB2 정보를 제공합니다.

http://www.ibm.com/software/data/db2/library/

DB2 제품 및 서비스 기술 라이브러리는 빈도 높은 질문(FAQ), 수정사항(fixes), 책 및 최신 DB2 기술 정보에 대한 액세스를 제공합니다.

주: 이러한 정보는 영어로만 제공됩니다.

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 로그인하십시오. /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 제품이 이러한 포럼을 통해 지원됩니다.

미국 외 지역에서 IBM에 연락하는 방법에 관한 정보는 **IBM Software Support Handbook**의 Appendix 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 국가의 IBM 공인 딜러는 IBM 지원 센터 대신 해당 딜러 지원 부서에 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® Universal Database SQL 참조서, 볼륨 1 및 볼륨 2 버전 7

책 번호: SA30-0997-00 SA30-0998-00

성 명		직위/담당업무	
회 사 명		부 서 명	
주 소			
전화번호		팩스번호	
전자우편 주소			
사용중인 시스템	☐ 중대형 서버 ☐ UNIX 서버 ☐ 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.

Printed in Australia

SA30-0997-00, SA30-0998-00